PROOF OF PURCHASE

Software Giveaway with Purchase of this Magazine
See back cover

HOME COMPUTER™ magazine

FOCUSING EXCLUSIVELY ON ● APPLE ● COMMODORE ● IBM ● TEXAS INSTRUMENTS

Vol. 4 No. 5

$3.50 in USA
$4.50 in Canada

A Ready-To-Use Quiz Construction Set for:

- Vocabulary Building
- Trivia Gaming
- Foreign Language Training

Arcade Gaming Action with Slithering Snakes and Birds of Prey

Your Personal Loan Calculator

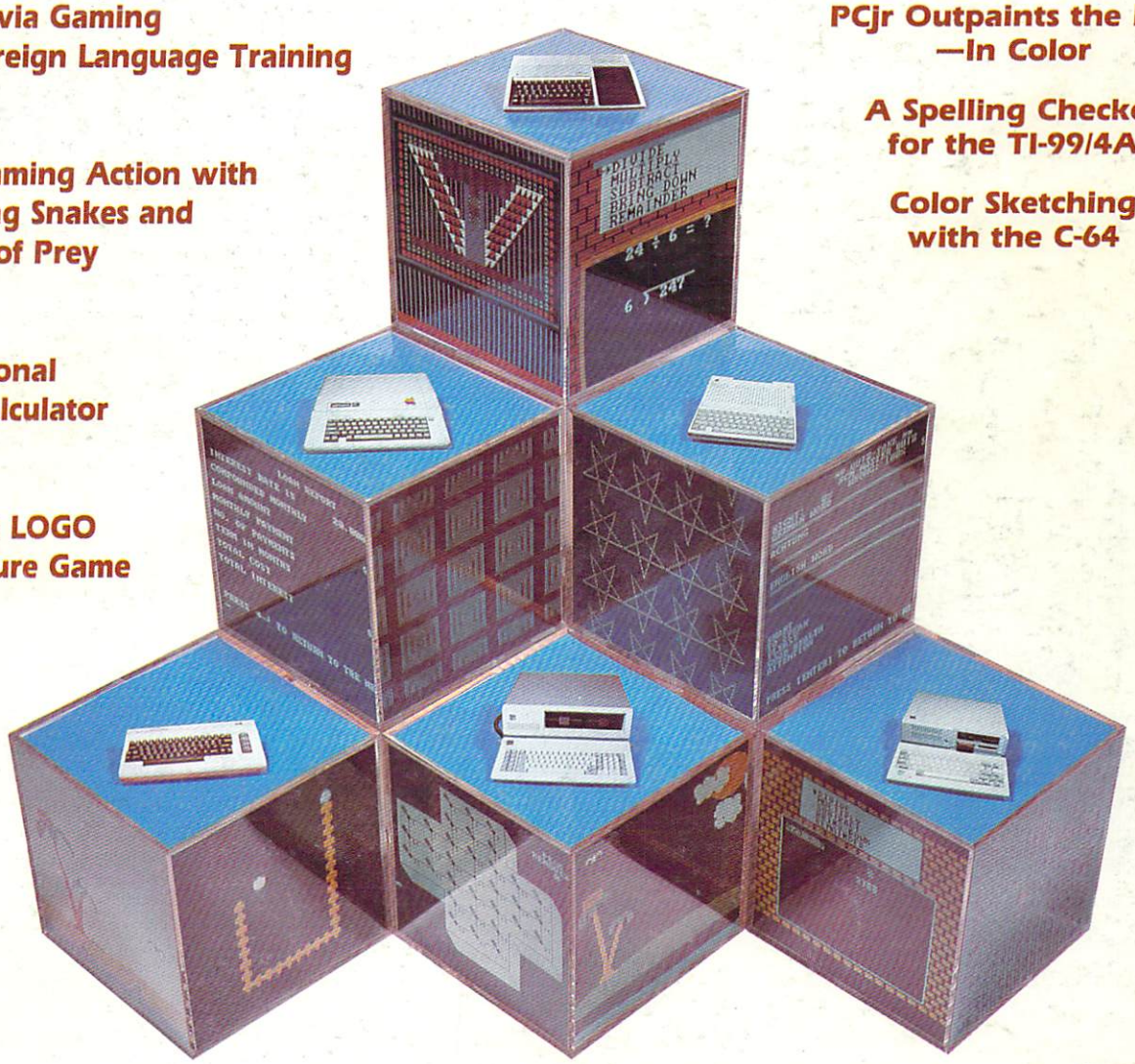
Building a LOGO Adventure Game

Secrets of Programming the Apple IIc

PCjr Outpaints the Mac —In Color

A Spelling Checker for the TI-99/4A

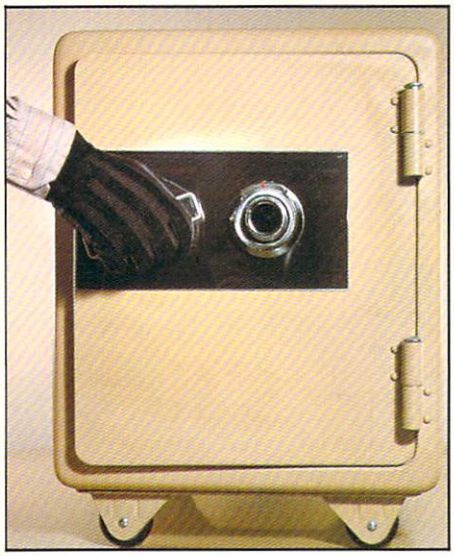
Color Sketching with the C-64

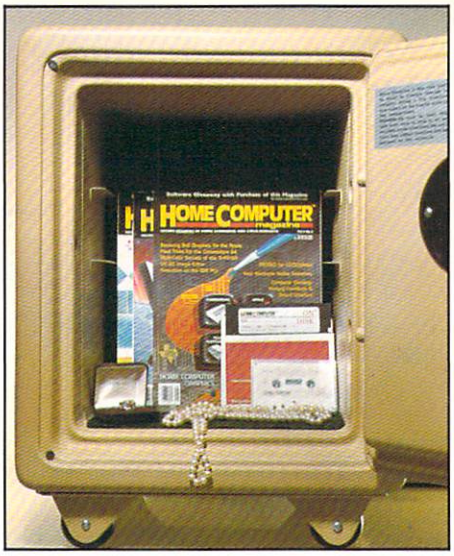


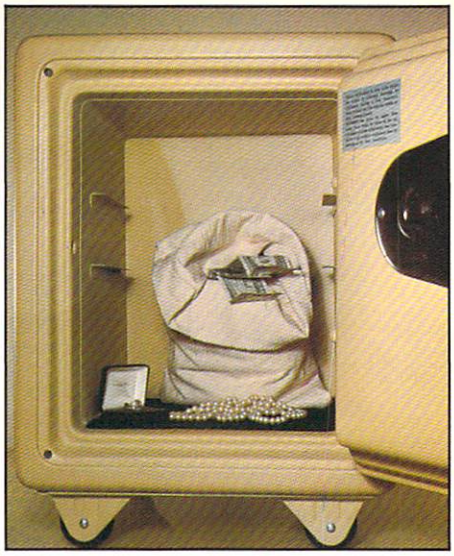
12509 05
0 477465

Building Up Your Software Library

Welcome New PCjr Owners: We Do Junior Right!



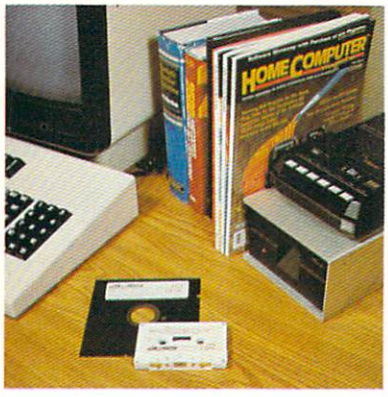




MISSING ANY VALUABLES?

If you're missing any back issues of HOME COMPUTER™ magazine you're missing more than you'll ever know . . .

Having each issue of *Home Computer Magazine* readily at hand provides you with direct access to a valuable reference library of home computer knowledge —unequaled anywhere!



A valuable reference library of each *Home Computer Magazine* issue is the One Essential Peripheral™ for your home computer.

Back issues of HCM's program service— ON DISK™ or ON TAPE™ are also available.

ON DISK™ and ON TAPE™ are the convenient, accurate and affordable ways to save hundreds of typing hours.

Collect all the programs from each magazine issue on a ready-to-RUN quality floppy disk or cassette tape available in separate versions for Apple, Commodore, IBM, and Texas Instruments home computers.

"Safeguard" Your Home Computer Knowledge— Order Valuable Back Issues Today!

To Order, Use Bind-In Card at Center of Magazine.

FOR NEW READERS

The Plain & Simple Truth About HOME COMPUTER™ magazine

Chock Full of Valuable Software & How-To Articles Without Filler

Every issue is a software "horn of plenty" with dozens of type-in-and-RUN programs printed in an easy-to-read listings format. Our programs are also available on inexpensive disks or cassettes for those who prefer the convenience of ready-to-RUN software. Step-by-step tutorials round out each issue, providing the solid facts you need without fluff or filler. Thus, each issue functions as an excellent reference work, as well as a valuable software source.

No Outside Advertising

Freed from the pressures of servicing *advertisers*, we concentrate on serving our *readers*. Each issue provides uninterrupted editorial flow and graphic layouts for better comprehension—plus unbiased product reviews which focus on true strengths and weaknesses, wherever the chips may fall . . . And we don't have to worry about losing advertisers because of publishing software in the magazine that is "too good." Consequently, we can provide the best free software available anywhere.

Focused on the 4 Hot Home Brands

We are 4 system-specific magazines under one wrapper—not a sprawling, "general interest" publication which attempts to cover too wide a field, only to spread itself too thin. The other side of the coin to this focused approach is the knowledge you gain from being exposed to the many tips, ideas, and techniques we provide for 3 of the 4 systems you may not even have. You'll learn more about your Apple, Commodore, IBM, or Texas Instruments home computer from this one magazine than from a host of more limited sources.

A Balanced Mix For a Perfect Recipe

In each issue we strive for a perfect balance of productivity, entertainment, education, utilities, and computer literacy—serving the needs of novice and pro alike. Every issue is a full-course meal, with a smorgasboard of tasty dishes for all palates. Whereas other computer magazines may dish out lumps of "editorial indigestion," we serve up a satisfying blend—one digestible byte at a time.

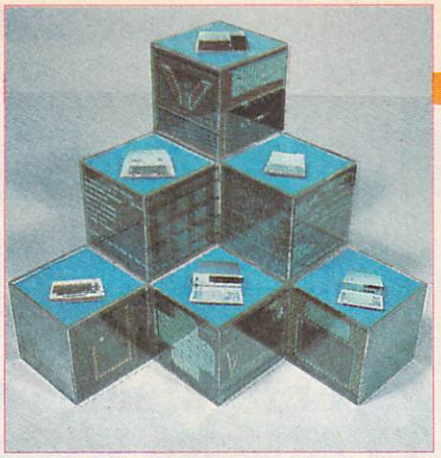
—Welcome to Our World of Home Computing

HOME COMPUTER magazine

Home Computer Magazine (ISSN 0747-055X) is published monthly by Emerald Valley Publishing Co., P.O. Box 5537, Eugene, OR 97405. The editorial office is located at 1500 Valley River Drive, Suite 250, Eugene, OR 97401 (Tel. 503-485-8796). Subscription rates in U.S. and its possessions are $25 for one year, $45 for two years, and $63 for three years. In Canada and Mexico add $11 per year. Other foreign countries $43 for one year surface mail. Inquire for air delivery. Single copy price in U.S. and its possessions is $3.50, and $4.50 in Canada and Mexico. Foreign subscription payment should be in United States funds drawn on a U.S. bank. Second-class postage paid at Eugene, OR 97401, and Columbia, MO 65201.

POSTMASTER: Send all address changes to **Home Computer Magazine**, P. O. Box 5537, Eugene, OR 97405. Subscribers should send all correspondence about subscriptions to above address.

Address all editorial correspondence to the Editor at **Home Computer Magazine**, 1500 Valley River Drive, Suite 250, Eugene, OR 97401. Unacceptable manuscripts will be returned if accompanied by sufficient first class postage and self-addressed envelope. Not responsible for lost manuscripts, photos, or program media. Opinions expressed by the authors are not necessarily those of **Home Computer Magazine**. All mail directed to the Editor or to the "Letters to the Editor" column will be treated as unconditionally assigned for publication, copyright purposes, and use in any other publication or brochure, and are subject to **Home Computer Magazine's** unrestricted right to edit and comment. **Home Computer Magazine** assumes no liability for errors in articles, programs, or advertisements. Mention of products by trade name in editorial material or advertisements contained herein in no way constitutes endorsement of the product or products by **Home Computer Magazine** or the publisher unless explicitly stated.



Outside HCM

Libraries have indeed changed. With your home computer, you now have the ability to house a vast library in a very compact space. But instead of shelves and books, the halls of *this* structure are lined with pixels and bytes. We picture on our cover the foundation of a great library—built not of piled concrete, but of original *software* piled high, and supporting the popular machines we cover. These are the actual programs available within the pages of this issue—a careful balance of entertainment, productivity, education, and utilities. So, as you start or continue to build your personal software library, we hope you'll turn to *this* Magazine for a readable supply of BASIC building blocks.

Inside HCM

Inherent in home computing is basic faith—a belief that the computer will make things better. Better work. Better play. Better education. And better creativity. At *Home Computer Magazine*, we have every reason to share in this faith because we've seen it *work*. We don't just *talk* about computers, we *use* them to make our magazine a better publication, and to help our readers become better served by their own machines.

According to psychologist Abraham Maslow, if the only tool you have is a hammer, you start seeing all of your problems as nails. Rather than provide just a "hammer," we instead open up an entire "tool box" for your use. For instance, take a look at *Quiz Construction Set*, a brain-building software tool with enough built-in flexibility to suit teachers, students, or any amateur self-educator. With this program-kit, you can construct quizzes for yourself or others on virtually any subject. (Just to get you started, we include three complete quizzes—courtesy of our "quizzical" magazine staff.) As an additional self-teaching aid, we offer a computerized *Division Tutor*. This program will exercise young minds in the old-but-still-valuable discipline of long division.

Many other valuable key-in-and-RUN programs appear in this issue of *HCM*, including the *Personal Loan Calculator*—another power tool to help you calculate answers you need without becoming too "amortified." Anyone even *considering* taking out a loan will jump at the chance to get ahead of the borrowing game with this practical program. And after you've jumped into *that* game, there's no reason to avoid *Jumping Ahead with Game Programming*—a lively program/tutorial with a ready-made game to boot.

For the artistically inclined, *Sketch-64* provides a handy little graphics tool—at the touch of a joystick—for Commodore 64 users. And if you're looking for pure enjoyment, try our two new game programs: scoop up schools of succulent fish with *Bird Brain*, or slide down a slippery path with *Slither*, the snake that snacks at every opportunity.

We have included our usual special features and how-to articles, like *Razzle Dazzle*, a small but dazzling bag of tricks for the TI-99/4A, and *Simon Sez*, a few good BASIC programming tips for the C-64. And for the benefit of all you new *IIc* programmers, we provide some hot clues for *Putting the Puzzle All Together: Apple IIc Programming Considerations*—an illuminating guide to maintaining program compatibility within the Apple orchard.

Orchards are fine, but you won't catch Junior standing under an Apple tree—especially when he's got some new keyboards to flaunt. There are, in fact, three new ones that we carefully scrutinize for you. Next, we doodle—in color—with a new, versatile artist's tool, *PCjr ColorPaint*. Finally, we cast off toward adventure . . . with a scintillating *Sailing* simulation and a colorfully animated *King's Quest*.

If your quest is to better understand new products before spending your hard-earned money this Holiday season, be sure to check out the rest of our reviews: You'll find such stocking stuffers as spelling checkers, speed reading courses, sketch pads, talking crickets . . .

And while the snow is drifting, you can be sifting—that is, sifting through a LOGO adventure which we start constructing in this gala issue.

So if you're seeking tools for improvement or bountiful activities for those cold nights ahead, you need look no farther than *Home Computer Magazine*.

Until next month, have fun reading, learning, and RUNing HCM

On Screen

By Gary M. Kaplan
Publisher & Editor-in-Chief

" . . . our reader mandate is crystal clear: continue to provide plenty of useful software, understandable "how-to" activities, and candid exposures of product merits and flaws."

The cold winds of January will be nearly upon us by the time many of you read this message. I'd therefore like to take this opportunity to wish all of our readers a joyous and healthy holiday season and a successful new year ahead.

With the advent of 1985, *Home Computer Magazine* will be entering its fifth publication year—a major milestone for us here in the Emerald Valley, the terminus of the old Oregon Trail. Maybe it's the land . . .maybe it's the people . . .but we like to think that the pioneering spirit never died out here. Perhaps that's why we're so excited to be blazing new trails with our recently implemented no-advertising format. With real pleasure, we can now bring you uninterrupted editorial flow and graphic layouts that fully support our technical content—conveying it as clearly and concisely as possible.

As evidenced by the many thousands of enthusiastic letters and phone calls, our readers overwhelmingly support this change. We're absolutely thrilled by your favorable response. For the future then, our reader mandate is crystal clear: continue to provide plenty of useful software, understandable "how-to" activities, and candid exposures of product merits and flaws. In 1985, our software content will be quite comprehensive. Already on the drawing board are several exciting productivity, educational, and entertainment packages that you'll be able to immediately type in and RUN, or load from our inexpensive floppy disks and cassette tapes.

By the way, I don't use the word "exciting" loosely. These software packages really *are* exciting because they provide a BASIC taste of the new, bestselling, state-of-the-art software that could set you back as much as several hundred dollars in a computer store. Obviously, our programs must necessarily be scaled-down versions of these big-name products, but for some, our programs will be all that you'll need to do the job—if, in fact, the job really needs "doing" in the first place . . . For others, however, experience with our software may whet some appetites for the far-more-costly commercial versions. Coupled with our product reviews, this hands-on knowledge will make you a wiser, more discriminating purchaser when choosing from a confusing horde of similar products.

In every facet of magazine and software publishing, we're striving for excellence—that is, to be the best at what we do. Being the *best*, however, takes ten times more effort than just being *good*. Consequently, it takes more time than a strict monthly schedule permits to prepare and debug each issue's software. Without cutting down on the sheer quantity of program coding in each magazine, we've found it to be impractical to put out twelve top-notch issues each calendar year.

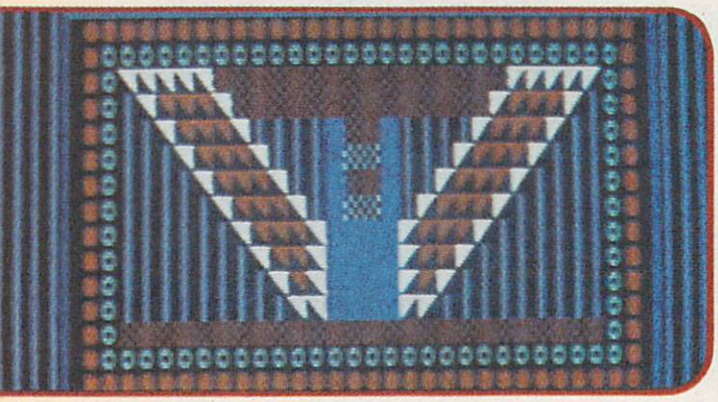
As a result, we've decided to solve this quality-vs-time dilemma by reducing our stated publication frequency to ten issues per year. Each issue will be identified by a volume and issue number on its cover and folio lines, rather than a stated month.

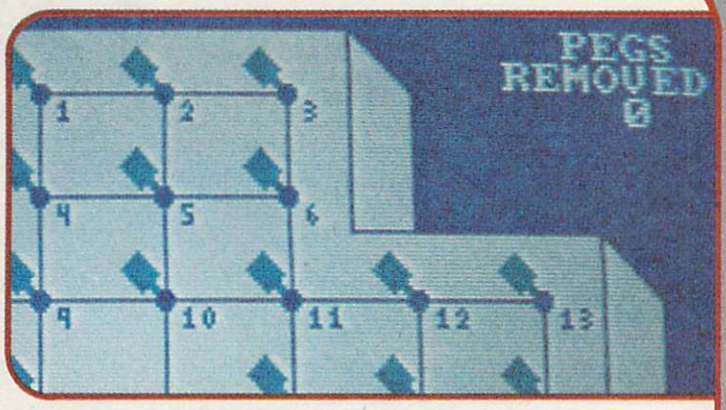
Present subscribers will still receive the correct number of issues they are entitled to based on their original subscription order. Renewals and new subscriptions entered prior to the cutoff date for publication of our next issue (Vol. 5, No. 1) will also be entitled to receive the magazine on this same 12-issue basis until expiration. All other subscriptions and renewals entered during 1985 will be based on ten issues per year, but new and renewing subscribers will still receive our free software media—ON TAPE or ON DISK—as premiums for joining (or rejoining) our "special family."

In closing, I'd like to thank all of you for your support and patience during this year of transition. Please keep your suggestions and submissions coming—it's the creative energy that fuels *your* magazine. And don't forget to introduce *Home Computer Magazine* to all of your friends, relatives, and associates who are interested in getting the most out of their present or future home computers. This kind of personal, word-of-mouth publicity is the best way to help us grow, while sharing with others the magazine you've come to value and trust.

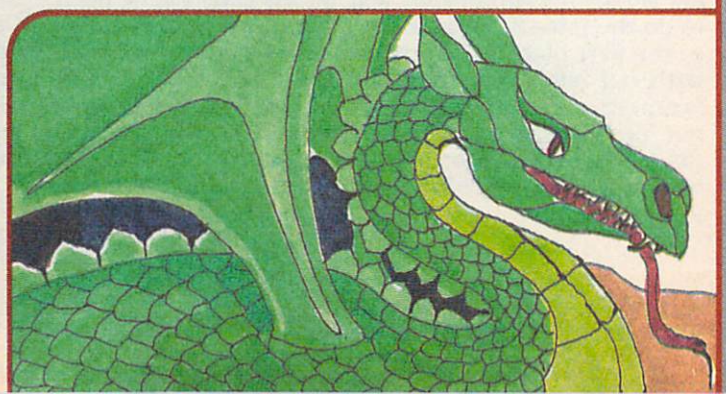
Thank you.

HOME COMPUTER™ magazine



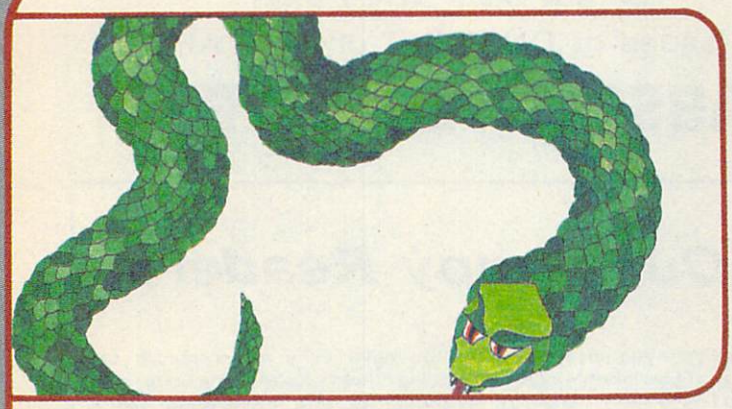


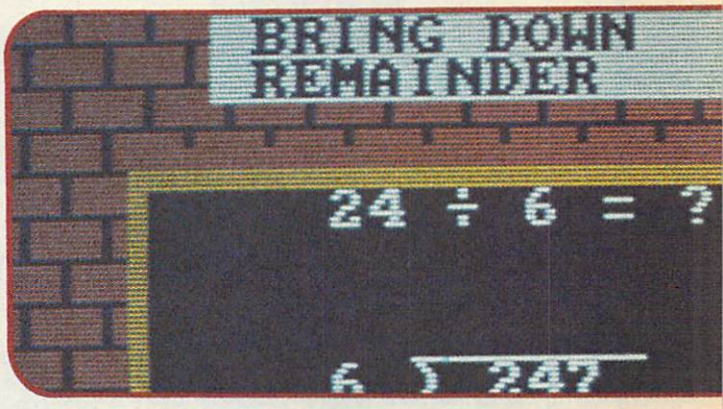
FEATURES



CONTENTS

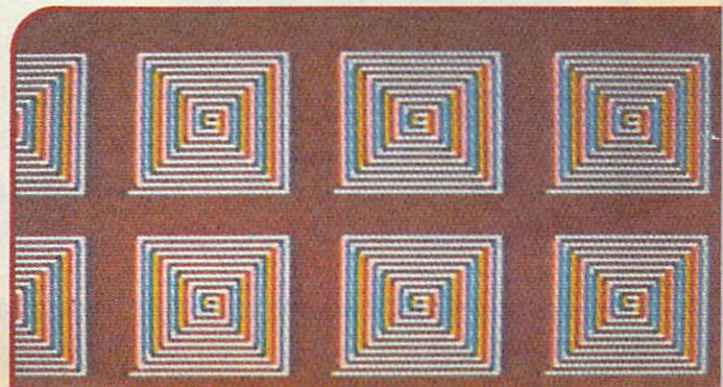
VOLUME 4 NUMBER 5





PRODUCT REVIEWS

DEPARTMENTS



MAKE SOMEONE HAPPY!

GIVE A GIFT SUBSCRIPTION TO

HOME COMPUTER™ magazine

As a *bonus* gift we'll send *YOU*
or the *GIFT RECIPIENT*
2 issues of ON DISK™ or ON TAPE™

ABSOLUTELY FREE!

The Perfect Gift For Any Occasion!

Comments From Our Happy Readers

"You broke new ground when you introduced your unique typesetting style for your program listings; well, you have done it again by separating all of the listings in the magazine into one section in the center of the magazine. I call that genuine brilliance!
Warren Agee, Livonia, MI

"Well it happened again. Your magazine arrived in the mail, and I'm completely delighted with it."
Chris L. Chaffin, Omaha, NE

"I have subscribed to your magazine since its inception. I must say it has been most informative and has provided me with answers to many of my questions. Your feel for what the public wants is uncanny!"
Larry A. Hamel, Millington, TN

"I just received your August issue. I ordered a 3 year subscription exactly 1 year ago, and I have seen it grow in size and quality. This latest issue, with the separate section of program listings, reaffirms my wise subscription investment."
Mike Oliver, Clarendon Hills, IL

"When I saw the new version of your magazine I was elated! Naturally I subscribed.
Doug Barker, Exeter, CA

"I was a former subscriber to the 99'er Home Computer Magazine and I thought it was great. Then when I got the first issue of the Home Computer Magazine, I was twice as happy. It was alot of information and great articles. Keep up the good work!"
Jenny Bures, Thousand Oaks, CA

"You have done a superb job of reaching other types of Home Computer enthusiasts and expanding your clientele while not depriving us 99'ers or leaving us by the wayside. The quality of the magazine is unsurpassed by any other, and I have looked at several different magazines! Hats off to you folks for your originality and continued endeavor to reach perfection."
John R. Stewart, Tucson, AZ

". . .I am extremely pleased that a magazine such as Home Computer Magazine is around. I find the magazine extremely well written and of invaluable aid. . .keep up the good work with the magazine."
James L. Grigsby, Richmond, KY

Thanks to your thoughtfulness . . . your friends, family, and associates can enjoy a gift that keeps on giving all year through! They'll enjoy

12 ISSUES* of HCM

delivered right to their door. . .
And you or the gift recipient will receive

2 issues

of our magazine program service

ON TAPE™ or ON DISK™

ABSOLUTELY FREE!!

Dozens of top quality key-in-and-RUN programs for Apple, Commodore, IBM, and Texas Instruments home computers appear in each issue. All of this for only $25*.

FREE software—ON TAPE™ or ON DISK™—the same high-quality programs published monthly in the magazine. This cassette tape or floppy disk program service—normally a $3.95 per issue* extra cost—is the convenient, accurate and affordable way to save hundreds of typing hours.

To Order A Gift-Subscription, Use Bind-In Card in Center of Magazine.

* Offer & Prices Subject To Change Without Notice.

Letters to the Editor

Snap-Calc Fan Mail

[Note: If you are a Snap-Calc user, please consult the "Debugs on Display" section in this issue for updates to all versions of Snap-Calc. Although the program runs fine, we've corrected some minor bugs and have added a few enhancements (including negative number inputs and results).

If you have a companion subscription to the ON DISK media service, the Vol. 4, No. 5 issue will contain these Snap-Calc updates in a "mergeable" file. For details, see the "DeBugs on Display" section on page 128.—Ed.]

Wants Larger Fields

Dear Sir:

I have enjoyed the last 4 issues of *Home Computer Magazine* tremendously. I have owned a TI-99/4A for the past 18 months and have found that your publication leads the field in 99/4A information. Congratulations on a job well done. Keep up the good work.

The program Snap-Calc in your August issue is a very well-written piece of work. I have spent hours "crunching" numbers with the program. I have also spent much time (to no avail) trying to change the program to delete the decimal point so that larger numbers may be input and processed. Could you suggest a change to the program that would allow operation with six place numbers? It would be greatly appreciated.

Mark Georges
Niceville, FL 32578

Mark, Snap-Calc is a large program and one that lends itself to programming errors when modifications are attempted. Changing the field size for the numeric entries would have a ripple effect throughout the entire program. We will look into this as a possibility, and if we can provide this change as a simple upgrade of the software, we will include the enhancements on a forthcoming ON DISK issue.

Grand Designs with Snap-Calc

Dear Sir:

As soon as I receive the diskette with your Snap-Calc program for my IBM PC, I intend to use it to produce accountings for the various trusts that I manage. Since I don't need as many columns as your program supplies, but do need to have the capacity to display a million-dollar transaction, I am wondering if the following changes would produce a row or column display of $1,234,567.

Line 210 . . . US $ = "#,###,###." . . .
Line 220 [delete last two "+US$+']

William Hovey
Boston, MA 02108

As stated above, William, the answer to increasing the size of the fields in the Snap-Calc program is not as simple as just changing the screen format—there are many considerations to be dealt with. But due to the interest expressed by many of the readers, our programming staff is looking into the viability of increasing the field size without a major redesign of the software.

Sample Now ON DISK

Dear Sir:

I typed in Snap-Calc only to learn that I couldn't understand what or how to use it. The documentation is confusing when you read the general description, then the specific machine version, etc. Perhaps you'd consider printing a sample Snap-Calc program which would give us dummies a chance (1) to see it typed in without homemade bugs, and (2) to get a rough idea of how it might be used.

I've enclosed my renewal for your always-improving magazine and hope to see it again delivered to my door, so I needn't waste computer time going to the BX for my copy.

Joseph E. Bowker
Bergstrom AFB, TX 78719

Joseph, anyone purchasing the Vol. 4, No. 5 issue of our Magazine's programs ON DISK will find two additional files: TRIPCOST and SEATTLE. The first file, TRIPCOST, contains the spreadsheet template or logic (to be loaded using option 2 of the spreadsheet load screen). The second file, SEATTLE, contains the data for the TRIPCOST template (to be loaded using option 1 of the load option screen). We hope that the availability of this sample template and data file will encourage many less-experienced users to explore the fantastic possibilities that our Snap-Calc program offers.

Macintosh Coverage Wanted

Dear Sir:

I congratulate you on the best magazine for the home computerist. My family and I started out with the TI-99/4A computer, and now have an Apple Macintosh with 512K memory, Imagewriter printer, Microsoft BASIC, etc. We use both machines daily. I hope you plan to include articles and programs for the Macintosh in your future publications.

Roswald J. Allen
Fitzgerald, GA 31750

You will be interested to know, Roswald, that we also "use both machines daily." All of our editors have a Macintosh as part of their technological arsenal. However, there are two factors that we must weigh when considering coverage of a new machine or language in the magazine: (1) Is there enough reader support for the machine/language to warrant adding the coverage? (2) Can we provide the added coverage without detracting from the machines/languages we presently include. In regard to coverage of the Macintosh, the time is not yet right.

Great Screen Photos

Dear Sir:

I read with great interest your sprite tutorial, "Double Your Color—Double Your Fun" in *Home Computer Magazine* Vol. 4, No. 2. I noticed that the illustrations used were actual photographs taken from a CRT screen and was wondering if you could tell me what camera and settings were used.

I was thinking of using photographs to record various graphic displays and any information you can provide on doing this would be appreciated.

Victor Meeldijk
Bronx, NY 10462

Thank you for the compliment on the photography, Victor. Those particular CRT screen shots were taken with a Mamiya RB67 Pro with regular Kodak color film. The display was a Panasonic composite color monitor, and the shots were taken in a darkened room with a light meter reading from the brightest portion of the screen. Tip: If the light meter flickers, we use a setting between the high and low readings. Also, never use a shutter-speed faster than 1/30th of a second. Good luck in your photographic efforts.

Bugs in Junior's Second Drive?

Dear Sir:

I have had a TI-99/4A system for three years and have two slimline drives installed in the peripheral expansion box. After working with my new IBM PCjr for awhile, I was ready to find a way to add a second drive.

The *HCM* Vol. 4, No. 4 article about adding a second disk drive to the IBM PCjr is just what I was looking for, but the MODBOOT.BAT file doesn't seem to work. I keep getting errors on the first and fourth lines. Is there a bug in the program?

The hardware modification went just fine. I found the chips called for at a local electronics store and bought a Panasonic double-sided slimline disk drive at another local store. I built the cable by carefully removing the connectors from an old TI disk cable and reusing them with a new length of ribbon cable. I sure hope you can help with the software problem.

Mark Guseila
Canton, MI 48188

Mark, you found two bugs in the typesetting of the article. On page 86, at the top of the second column, the MODBOOT.BAT file is described and then listed in bold type. The first line of the listing reads: AO O:980, *but should read:* A 0:9080.

The second error occurs in the fourth line of the same listing, which reads:

OR BY (410),40

In fact, it should have brackets, not parentheses, like the following:

OR BY [410],40

We hope you enjoy your dual-drive system. Here at HCM we have several PCjrs configured this way.

Continued next page

C-64 Memory Trick or Treat?

Dear Sir:

I have found that if I enter POKE 644,255:SYS 58260 on my Commodore 64 and press [RETURN], the start-up message changes from 38,911 to 63,231 bytes free. That's 24K of extra memory! It seems too good to be true, but PRINT FRE(0) returns a value of -2307, which shows that the extra memory is really there. Even typing NEW cannot get rid of it. Does this mean that I can now write BASIC programs that are up to 62K long? What am I POKEing and SYSing to, anyway? Could it be the hidden RAM behind the ROM I hear so much about?

James Redd
Camden, OH 45311

You're right, James, it is too good to be true! What you have changed is a pointer used by the system, but you have not added any memory to the system. The BASIC interpreter will get confused and cause the computer to "crash."

Apple *IIc* with TI Printer

Dear Sir:

I am considering buying the new Apple *IIc* and have some questions you might be able to answer. Can I use my TI impact printer (Epson MX-80 with serial card) with the Apple *IIc*? If so, are any modifications required? Can I use the TI disk drive (Shugart 400L) from my TI expansion box as an external drive for the Apple *IIc*? Are any modifications required? Can I leave the drive in the expansion box, disconnect it from the TI controller card and connect it to the Apple *IIc* external drive port, then power up the expansion box to run the drive?

Any assistance you can provide will be greatly appreciated.

Jack A. Sharp
APO San Francisco 96301

The answer to your first question, Jack, is yes—you can use your TI printer (with the serial card installed) connected to the Apple IIc. We have done that here with no problem at all. Adding a TI disk drive (Shugart 400L), however, is not possible. The Apple II drive signals are not compatible with the disk drives used by the TI disk system.

To learn more about the IIc, Jack, we recommend that you read the article in the previous issue of HCM entitled "IIc: The Core of a New Machine" as well as its follow-up article in this issue "Putting the Puzzle All Together: Apple IIc Programming Considerations."

Needs Software for C-64

Dear Sir:

I am having difficulty finding "in-depth" literature for programming my Commodore 64. Retail store salesmen cannot help, and the one computer store selling Commodore 64's wanted to sell me hardware and software before finding out what I was really trying to program.

Briefly, I am an engineer engaged in estimating for a manufacturing and machine tool company. Armed with charts and a desktop calculator, I estimate machining times, horsepower, and thrust on special machines. The "number crunching" ability of the computer would enhance my job enormously. The "hard copy" via a printer is invaluable to engineers, once we are awarded a contract for a machine(s).

My specific need is for a clear way to program an array and be able to pick out the information (by row and column), desired for subsequent use in my program. Printing the array on the screen or a printer is a waste—I need to pull only one piece of data out of memory and use it. Searching charts is what I am doing now.

I would appreciate your help because very little business programming is advertised for this computer.

John R. Johan
St. Clair Shores, MI 48081

John, we're not sure that we understand exactly what you're asking for. It sounds like you're looking for a data base manager for the Commodore 64 that will pass information into one of your own programs that you're using for doing calculations. This is not a "trivial pursuit." We do not have an immediate answer for you. Perhaps some of our HCM readers who have done something similar can help by making a recommendation for the appropriate software.

A User-Happy Letter

Dear Sir:

I do not wish to belabor the obvious. This is the most user-friendly magazine that I presently subscribe to that covers computers.

To show my enthusiasm, I have bought the *Best of 99'er* book and tape package and all of the back issues available.

I am glad that your efforts to make this magazine the best of its kind are succeeding.

Alexander Jaffe
Highland Park, NJ 08904

Thank you for the kind words, Alexander, and we appreciate your enthusiasm. For you and other readers with a TI-99/4A, we now have all of the software from the 99'er HCM back issues available on magnetic media—both ON DISK and ON TAPE. Besides being a real great value and a terrific way to expand your software library, magazine and media back-issue sales also go a long way in supporting your favorite magazine—a publication that is (by choice) devoid of outside advertising.

Wanted: C-64 Slave for 99/4A

Dear Sir:

I own both TI-99/4A and Commodore 64 computers. After many hours of programming with both machines, I have come to the conclusion that the TI-99/4A is a much superior machine to work with.

The TI has a Terminal Emulator and the Commodore has the Super Expander. While I concede that the Commodore has more memory and runs faster, programming with it is a chore.

If at all possible, I would like to utilize the Commodore as a "slave" unit to the TI-99/4A. Is this feasible, or should I put the Commodore on the shelf and forget the whole thing?

I would like to be able to have the TI machine access the Commodore's memory capacity. Is it as simple as connecting an RS232 cable between the two machines?

Robert W. Folsom, Jr.
Peabody, MA 01960

You pose a very interesting question, Robert. At this time we don't know of anyone who has used the Commodore 64 as a slave to the Texas Instruments machine. We agree with you, it would not be as simple as hooking up the RS232 cable between the two machines. The RS232 port is actually under the control of the processor in each computer, and in order for one processor to use the other computer's memory, the second computer's processor would have to be disabled. It should be possible to build an adapter to allow Direct Memory Access of the C-64's memory from the TI-99/4A main bus connector (on the right side of the console). This would require special logic to be designed and fabricated along with a special cable. We are not aware of the existance of such a device. But, this might be an interesting project for consideration—any adventurous spirits out there?

Tech Literature for the *IIc*

Dear Sir:

Fine magazine—keep up the good work. I hope you can help me. I have an Apple *IIc* and am looking for an *Apple IIc Reference Manual*. Do you know if any exist and where I might obtain one? Also, do you know of any books on assembly language programming of the 65C02, which is used in the Apple *IIc*?

Karen M. Lee
Westbrook, ME 0409

Karen, I think that the answers to your questions are probably contained in this issue's article on the IIc entitled, "Putting the Puzzle All Together: Apple IIc Programming Considerations" (a follow-up to last issue's article "IIc: The Core of a New Machine").

Words From South Africa

Dear Sir:

I found the back issues of *HCM* to be most valuable in getting MULTIPLAN and TI-Writer together. I have a FACIT 4510 printer on-line with the TI-99/4A. The FACIT 4510 is a quality matrix printer currently selling at $1070 from some shops. The magic password (device name) which was discovered by a genius I found in Benoni, is

"RS232.BA = 9600.TW = 1.CH = N"

Incidentally, I work for Gillette South Africa, Ltd. as manager of their Affirmative Action Plan. We are a Sullivan signatory company. Once a year, a lengthy report is sent to the States to the Reverend Leon Sullivan in order to negative the disinvestment campaign being waged against American companies in the Republic. The Sullivan Report has to be audited for expenditures on our social responsibility activities. In May of this year I started using MULTIPLAN to record a year's expenditures. The resulting spreadsheet, all nicely laid out and pasted after printing, resulted in saving the auditors four days of work when compared with the time taken to audit the 1983 report. How is that for the TI-99/4A? This use of MULTIPLAN is a first in the Republic. You may wish to include this information in *HCM*.

G. E. Bagley
Republic of South Africa

Thank you very much for the infomation. We're sure that readers who have the Facet 4510 printer will be interested in learning of your "magic password."

C-64 to TI Printer—The Missing Link

Dear Sir:

In your Vol. 4, No. 4 issue of *HCM*, you gave Edmond Reynolds only half the answer on the method of hooking up a C-64 to his TI-99/4A printer. You are so right that the Cardco interface is necessary, but as I found out through much pain and suffering, that's not all folks.

Only after hitting every software store in my area and making "dozens" of frustrating long distance telephone calls to TI and Epson (after all, the TI printer is an Epson MX80 inside) did I finally get the answer from a TI technician.

In order to use the printer as a Centronics parallel, you must remove the serial board (the one containing the serial port) from your printer. To do this, follow the instructions in the TI printer manual for getting to the lower dip switches in the machine. Just remove the four screws holding the board down and disconnect the one wire hooked to the serial board. Gently remove the board (it's plugged into the lower board) and leave it out with the one loose wire hanging.

What you now have is a Centronics parallel Epson MX80 for use with your Cardco interface and the C-64 computer.

It's real easy to do, but what a job getting the information.

Allen Palazzo
Staten Island, NY 10304

You're right, Allen. We forgot to mention that in order to complete the hook-up of the Epson printer for parallel operation you need to pull out the serial board.

Apple Cassette Support On Wane

Dear Sir:

I am thinking of buying an Apple-compatible clone (I have been offered a new one at a good price). However, initially I won't be able to afford a disk drive.

I foresee my uses of the computer as being educational (for my five-year old son), entertainment (games, etc.—especially adventure games), probably some home management software, and a little experimental programming.

My main concern at the moment is whether or not Apple software is available on tape? If I type in programs from magazines, can I save them on tape? Are there any restrictions on doing this? (Do some programs look for a disk? If so, how can I tell?)

Thank you for any help you can give me.

H. King
Cornwall, Ontario, Canada

Apple has been phasing out its cassette software support for the Apple II Family. In fact, the Apple IIc doesn't have the software in its ROM to support a cassette port at all. Cassette-based software for the Apple II Family is very rare. When selecting an Apple program to key-in from Home Computer Magazine, read its accompanying article to see whether it requires a disk drive. The following articles in this issue include programs that can be used with a cassette, without any modifications: Personal Loan Calculator, Jumping Ahead With Game Programming, Division Tutor, Bird Brain, and Slither. Demand for Apple software on cassette is too low to warrant ON TAPE production.

Finding a TI p-Code Card

Dear Sir:

After spending $30 in phone calls and about $6 in postage, I need your help. Having subscribed to *HCM* (and *99'er*) for 2 years, your loyal readers are my last resort. I've nowhere else to turn.

I'm looking for the Texas Instruments p-Code card (UCSD Pascal). I've written to more than three-dozen supply houses and listed my name on dozens of bulletin board systems across the country, but *no one* knows where I can purchase the card.

If anyone has a p-Code card and/or the supporting cartridges or software available, I'd like to hear from them. This TI owner would like to complete his system, but even Texas Instruments, Inc. won't tell me *who* bought the remaining stock of Pascal software or the p-Code cards.

Don Graff
Byhalia, MS 38611

We have learned that the p-Code card and all the UCSD Pascal software packages are available, but in short supply, from Tex-Comp (P.O.Box 33084, Granada Hills, CA. 91344). The complete package price is $119.95 (including software), and the firm requests that you order this particular package by mail only.

Junior Monitor Questions

Dear Sir:

The Vol. 4, No. 2 article on the PCjr mentions that to obtain screen mode six (high resolution), you must have a compatible direct-drive RGBI monitor. I do not understand the difference between this monitor and composite color monitors, and I was wondering if the *HCM* staff could give me any help.

I would like to comment on your wonderful magazine. I love the reviews and the program listings. You have the best magazine I have seen in a long time.

Mark Guebert
Arnold, MO 63010

Mark, the three picture-tube color guns in the RGBI (red, green, blue, intensity) monitor are driven directly from computer signals. This allows detailed graphics to be displayed. A reasonably good RGBI monitor can be used to display 80 columns of text along with very good color graphics. In addition, the RGBI colors are "pure" in appearance.

A composite color monitor takes a single input of serially-coded (usually refered to as the "NTSC standard") information. This information contains the intensity signals for each of the three colors and is fed to the monitor in a serial fashion (red information followed by blue, followed by green, followed by red, etc.). All this must be decoded by the composite monitor and then used to activate the color guns in the picture tube. This method of transmitting information to the monitor makes it difficult to do precise positioning of color dots on the screen, causing "smearing" of the colors at points of color change.

TI Program Recovery

Dear Sir:

I have discovered some useful information for TI owners. Have you ever been typing in a long program and then accidentally pressed QUIT before you had a chance to save what you were typing? Well, if you have Extended BASIC and the 32K memory expansion, you can easily retrieve your lost program.

Here is an example of how to do this: First type a short program in Extended BASIC with the memory expansion on. Then type CALL PEEK(-31952,A,B,C,D) :: PRINT A;B;C;D The first two values shown point to the start of the line number table. The second pair of values point to the end of this table. Write down these numbers. Now press QUIT and re-enter Extended BASIC. Next, type CALL INIT Now you have to reload the values that you wrote down. Do this by typing CALL LOAD(-31952,W,X,Y,Z) replacing the W with the first number you wrote, the X with the second value, etc. Type LIST and presto! You have found your program.

There is only one drawback. If you add any lines to your program after you have found the four numbers, you must re-PEEK that address and get the four new values. If you do not do this, and then try to place the old values into memory, your computer will most likely lock up. I hope you find this information useful.

Mark Finkelstein
East Windsor, NJ 08520

Continued

Say, that's truly an interesting technique, Mark. It is one that can come in very handy when trying to debug a large Extended BASIC program. Thank you for the information.

Taking the C-64 for a Texas Drive?

Dear Sir:

I am writing to you to ask if there is any possible way to hook up our Commodore 64 computer to our Texas Instruments disk drive? If so, how and where can we purchase the items needed and at what cost? We would appreciate an answer as soon as possible. We subscribe to your magazine and would like to say keep up the good work.

Susan Jaszkowski
Columbia, SC 29209

Sad to say, Susan, the Commodore 64 disk drive contains its own built-in disk drive controller and therefore is not compatible with the 99/4A disk drive which requires a separate external disk drive controller. If any reader knows of a disk drive controller for the C-64 that will work with SA400-style drives, please write to us.

HCM Wins Award

Dear Sir:

Congratulations! After nearly a month of comparing *HCM* to all the other computing magazines, I hereby proclaim you the winner of the first annual D. J. Branham Computer Magazine Contest!

As the grand prize winner, you will receive my subscription for two, not one, but two full years. In addition, I am also awarding you my subscription to ON DISK for 12 months.

This contest was conducted using my Apple *IIc*. This decision is final, as long as you continue to put out a truely superb magazine.

Seriously though, your magazine is far better than other magazines, even those that specialize in only one brand of computer. In future issues I would like to see you review and list entertainment and home management software which utilize the Apple mouse.

Thanks for a superior home computing magazine. Keep it up!

Danny J. Branham
Lawton, OK 73501

Gee thanks, Danny! We must be doing some things right—we are receiving lots of similar awards from all over the globe. We will watch for mouse-related software to bring to you, both in reviews and as an option in future HCM programs.

European Color TV Problems

Dear Sir:

I am a subscriber to your magazine and also an owner of a TI-99/4A computer. Your magazine is a great source of information for the home computer field.

I would like to ask you for a solution to one problem I have with my computer. I can't get a color picture on the screen. I have the U.S. version TI-99/4A which produces a composite NTSC video signal with 3.58 MHz color carrier. I bought a color TV set (SONY KV 1423ME3) which is able to pick up PAL video signals with 4.43 MHz color carrier. I found in the circuit block diagram of this machine that the video signal is generated by the TMS9918A video processor chip. This chip is driven by a 10.7 MHz quartz resonator, and inside this chip is the new signal with a frequency of 3.58 MHz, which is probably the color carrier 3.58 MHz.

My question is: Can a replacement of a 10.7 quartz resonator be made for a resonator with higher frequency—which would produce the color carrier of 4.43 MHz—or must the TMS9918A chip be replaced by a TMS9928 chip, which is used in the European version of the TI-99/4A? If yes, which components must be replaced?

I am considering buying an IBM PCjr computer—also U.S. version. I should like to get information on any simple arrangement for operation of this machine in NTSC 4.43 that can be made.

I think that my problem described above may be interesting to many owners of U.S. versions of personal computers outside the USA.

Thank you in advance for your answer.

Pavel Strihavka
Prague, Czechoslovakia

The TMS9928 used in the European model of the TI-99/4A interfaces to the video through encoder circuitry which includes its own 4.43 MHz crystal for compatibility with the PAL standard. The TMS9918A VDP found in US models of the TI-99/4A has the circuitry onboard the chip for producing the NTSC composite video with the 3.58 MHz color carrier. Both of these TI-99/4A Video Display Processors use a 10.7 MHz crytal for timing and should not be changed. We fear that your NTSC-compatible TI-99/4A is unable to interface with the Sony TV that you describe without major modification. If you did replace the TMS9918A with a TMS9928, you would still need to add the PAL composite video encoder circuitry as well. The situation with the IBM PCjr is about the same—we recommend that you purchase one designed for Europe.

He Finally Found Us

Dear Sir:

Several years ago when I first purchased my TI-99/4A, someone told me what a great magazine the *99'er* was. After months of diligent searching, I finally gave up hope of ever finding a copy.

Imagine my amazement as I stood in a major department store trying to buy a microwave oven and there on the counter in front of me is this *Home Computer Magazine*, and in this little bitty Texas map it says "Continuing *99'er Magazine*'s coverage of the TI-99/4A." I snatched up both issues that were lying there and have been thoroughly engrossed in them ever since. My wife is still waiting for me to set up her microwave!

You have a marvelous magazine and a new dedicated reader.

J. David Schronce
Chicago, Ill. 60611

We're very glad you finally found us, David, and hope you enjoy your subscription. We also hope that word-of-mouth from you and other readers will get the message out that we are still "cooking." Now, why not be a good spouse and set up that microwave!

Getting a Jump on Junior

Dear Sir:

I'm writing to you regarding the article in Vol. 4, No. 4 of *Home Computer Magazine* about adding a second disk drive to the PCjr. I went through the procedure twice and both times ended up with the same result. I'm hoping you can help me.

I have a new IBM PCjr and a Percom 5-1/4" external disk drive (I believe that it is a Tandon 100-2). The literature that came with the drive doesn't show the jumper position for configuring it as a drive B:—this may be part of my problem. When I connect the new flat cable and turn on the power, the disk spins no matter which way I turn the cable (the red light comes on only momentarily). However, when I turn on the computer, it stops spinning. Also, the computer will not read drive A: (on power up, it reports ERROR H). If I do get it to read drive A: (by pressing ENTER after ERROR H), it doesn't function properly. It starts reading a file and then reports a sector-not-found error.

Mark Beifuss
Bryan, TX 77801

Mark, from your description it sounds like you should check two items, both of which are in the Tandon 100-2 disk drive: (1) Make sure that the load resister IC pack has been removed from the drive (this can cause ERROR H problems). This load resistor pack is located near the ribbon-cable connector and is usually the only IC that is inserted into an IC socket. (2) When accessing drive A:, notice whether the external drive's red light also comes on. If so, you still have not found the right address selection as described on page 85 of the article.(If both drives are accessed at the same time, a "sector-not-found" error will most likely occur.)

Port-to-Port on the C-64

Dear Sir:

Is it possible to change the C-64 device number of the printer port with a POKE, or something from #4 and #5 to #2, so that an RS232 serial printer could be used with some of the canned software that do not offer this option?

Can the Commodore 64 computer be run and programmed from a remote terminal through the RS232 port? Do you know of a POKE statement or simple program that would allow this?

Scott Schultz
Athens, GA 30606

No, Scott, the hardware design of the C-64 will not allow the redirection of the "canned" printer output functions to the RS232 port. The answer to your second question is essentially no, but for clarification read the answer to the related question from Mr. Folsom (see previous letter, "Wanted: C-64 Slave for 99/4A").

Never Too Late for Software

Dear Sir:

Several comments I would like to make:

No advertising—Wow! Did not think this could be done in a computer magazine. I think the letter from Sandy Foote [Vol. 4, No. 4] proves your point well.

Most of your assembly language programs seem to involve the Mini Memory module rather than the Editor/Assembler. How about more parallel listings?

My only real complaint is that I did not discover your magazine sooner, particularly when it was strictly for the TI-99/4A.

Donald L. Mahler
Newton, MA 02159

We're glad you approve of our new format, Donald. Editor/Assembler source code listings for all those old Mini Memory programs are now available ON DISK for past issues. These disks also have the Mini Memory object code files. By the way, you don't have to feel sorry that you didn't discover us sooner because "The Best of 99'er—Vol. 1" and later back issues are still available—while supplies last. See the center bound-in order card and additional information on the inside rear cover.

Commodore Ribbon in the Black

Dear Sir:

Reading about the trouble your readers are having in locating printer ribbons for the Commodore 801 printer prompted me to write about how I solved the problem.

I bought an inkpad inker from my local office supply store. Whenever necessary, I re-ink the felt pad of the printing mechanism. If you ink the felt pad too much, the first couple of lines will be too dark. This darkness will soon disappear.

Hugh A. Valliant
Toronto, Ontario, Canada

Thanks, Hugh, for the tip on the Commodore 801 printer ribbons. It sounds a little messy, but it should extend the life of a few ribbons.

Is Junior's a Single or Double?

Dear Sir:

I have a question that no one thus far has been able to answer for me. I have the double-sided 360K disk drive on my PCjr with which I use both single- and double-sided disks. Sometimes when I either DIR or format a single-sided disk, I'll receive a message indicating that I have the byte capacity for a double-sided disk! However, with some brands of single-sided disks, I'll get the correct capacity of 180K.

How does the double-sided disk drive "know" or distinguish between a single- and a double-sided floppy disk?

My best wishes to you on your new magazine format. It sounds like quite a challenge to omit all outside advertising. I sure hope you succeed. We've all seen a lot of computer magazines go under this year. Yours is the only one left that gives fair space to the PCjr. Keep up the good work.

Brad L. Barnes
Redwood City, CA 94063

You may purchase a diskette that is "rated" as a single-sided diskette, Brad, but if you format it yourself without specifying that it be formatted as a single-sided diskette, the PC-DOS FORMAT command will automatically default to a double-sided format. Most diskettes—whether they're marketed as single-sided or double-sided—will work as double-sided diskettes.

The PCjr PC-DOS 2.1 records what is called a "boot track" in the very first part of track 0 on each diskette as it is formatted. Information recorded in this area includes the format of the entire disk (single-sided, double-sided), the number of sectors it has per track, etc. This is the information accessed by the DIR command.

Tech Note Request Granted

Dear Sir:

In Vol. 4, No. 2 you had a most desirable and useful page in the "Tech Notes" section—the page on the TI-99/4A that was written by William K. Balthrop. Is there any way you can get some more of these unusually valuable ideas from him?

Specifically, I might mention the desirability of a basic routine that would take the place of the Extended BASIC command +LINPUT+.

Hope to see some more of his ideas.

Joel Martin
Fort Lauderdale, FL 33314

Ask and ye shall receive, Joel. See this month's TI-related "Home Computer Tech Note," which describes how to generate a LINPUT statement in TI BASIC.

Reading Format Enjoyable

Dear Sir:

I just finished reading the latest issue of your magazine in its new format. You have taken a bold and courageous step to break out of the traditional mold, and I for one applaud you for it. For the first time ever, I was able to read a computer magazine all the way through and found myself enjoying it because of the lack of clutter and the ease of continuity—much like seeing a good TV program without the annoyance of commercials.

You have a great magazine and you have just made it better. Keep up the good work, keep serving the TI user as long as possible, and good luck in a tough environment.

John F. Banocy
Englewood, FL 33533

Thanks for the vote of confidence, John. We especially like the TV analogy—it made our publisher feel like Alistair Cook on PBS's Masterpiece Theatre.

Apple User Likes It "Jam-Packed"

Dear Sir:

I am an Apple user who is very pleased with your magazine. You have many programs in an easy-to-follow format and with no outside advertising, this publication is superb!

I am interested in starting a software exchange program for Apple. I would appreciate it immensely if you could help me out by either writing to me or by printing this letter in your magazine.

One last word. Your magazine is jam-packed full of useful information and is one of the best in the field.

Thank you.

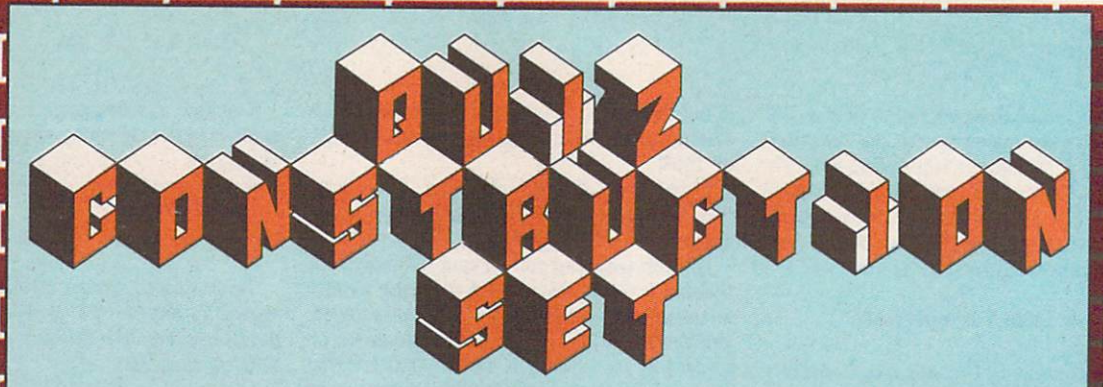
Mike Huston
Mendon, MI 49072

We appreciate your support and praise of Home Computer Magazine, Mike, and will do our best to continue to fulfill your expectations. One caution to you and others who start software exchanges: Beware of copyright infringements on the software appearing in Home Computer Magazine or in other magazines. Our very existence depends on reader support for software sales of ON DISK and ON TAPE. The software provided in Home Computer Magazine is protected by copyright and is not public domain software. Be very careful that any software put into your exchange is in the public domain. If you'll send us details on how the exchange will operate, we'll see whether we can help you get started. Good luck in your endeavors.

HCM

Special Announcement:

Home Computer Magazine is looking for "One-Liners". If you have written a 1-line program in any language that is available on the computers we cover, send it in addressed to *Letters to the Editor*. It may win a prize! Now turn to page 34 to enjoy the best of this issue's One-Liners.



by William K. Balthrop
HCM Staff

***Calling all teachers, students, trivia and non-trivia buffs—
All who seek self-improvement and greater knowledge . . .
Create your own questions and find your own answers
with this do-it-all quiz machine!***

In the beginning, there was the question. Then came an answer—and the first quiz was born.

Many questions—and answers—have resulted from mankind's sometimes trivial, sometimes not so trivial pursuit of information and knowledge. Besides asking the eternal questions common to every generation, people are devoting more and more of their time to educating themselves in every area of human interest. And, at work or at play, the basic question and answer quiz is still a favored learning and teaching aid.

A quiz may be spoken or written—or nowadays, designed, stored, and taken on a home computer. Just a few years ago, the "teaching machine" was pretty much a joke—a complicated electro-mechanical device stuck in some school lab, and probably bolted to the floor. Now, teaching is but one natural function of a much smaller, multi-purpose device. With your computer and the program published here, you can do everthing you could have done with that bulky old machine—and much more.

Quiz Construction Set is just what it says: a program that provides all the pieces you need to build, store, and retrieve your own direct-response quiz. It is perfect for school or home learning situations—and can provide a good deal of entertainment as well. You may enter questions of virtually any type, on any subject, with accompanying answers. Use them to exercise your own memory or someone else's. You also may select one of two types of clues to accompany a question, and determine how many chances will be given to get the right answer. As you take the quiz, the program keeps a running score of both right and wrong answers—and also checks your answers for correct spelling, tabulating a score for that as well. (For more on spell-checking, see "Taking The Quiz," page 17.)

Two In One

This software is actually two programs in one package: *Quiz-Make* and *Quiz-Take*. Here again, the names tell you what to expect. You may use the first routine to construct and store your quizzes, and the second to retrieve and take them. This structure serves several purposes: First, it frees up memory to hold larger quizzes. Second, it offers a form of security: If you don't want the quiz takers to be able to modify the quiz, you can give them only *Quiz-Take*, which has no provisions for making alterations in either questions or answers.

Quiz-Make

Use this program to create and modify a quiz before you use *Quiz-Take*. The type of quiz you create is limited only by the total number of questions, your system's memory capacity, and the size of the question and answer fields. The size of these fields is limited to two screen lines for a question, and one screen line for an answer. The maximum screen width is 40 characters on the IBM, Apple and Commodore computers, and 28 characters on the TI-99/4A.

Let's go through the process of setting up a simple quiz. After loading and running the program, you will be shown a title screen. To progress to the Main Menu, press either (ENTER) or (RETURN). You will see six choices:

1) EDIT
2) LOAD
3) SAVE
4) PRINT
5) CHANGE PARAMETERS
6) EXIT

To start, press 1. If you were modifying an existing quiz, you would simply begin entering your new questions and answers. Because this is a new quiz, you will be taken to the parameter setup screen and asked to design your quiz:

QUIZ TITLE — Enter the title of the quiz. This title will be displayed on the top of the screen during the Edit mode, and while a person is taking the quiz.

AUTHOR'S NAME — Enter your name here if you are the quiz's creator.

QUESTIONS HEADER — Enter the prompt you would like to see above all of the questions. This could be the name of a category, or simply the word "Question."

ANSWERS HEADER — Enter the prompt you would like to display above the answer field.

```
QUIZ TYPE: 1. SEQUENTIAL
           2. RANDOM
```

If you press 1, the Sequential option, the quiz will be given in the *same* order that you enter the questions. Option number 2, Random, means that the questions will be selected at random from the entire quiz file.

PERCENTAGE OF LETTER CLUES (0 - 80) — This option determines how many letter clues will be given for a missed answer. When the Letter Clues option is selected in the *Quiz-Take* program, the student is shown a few of the letters from the answer. The number of letters given is calculated by multiplying the letter-clue percentage times the total number of letters in the correct answer. The spaces where letters are not displayed are filled with asterisks. A 50% letter clue might look like this:

INTERPOSITION (correct answer)
TE*P ITI** (letter clues)

It's possible that fewer clues than the percentage you selected will be displayed. This will happen if the program chooses the same letter twice. In the example above, if the program had twice picked the first T in the word INTERPOSITION there would then be one less letter clue displayed. You should think of this option as a *maximum* letter-clue percentage.

"You may enter questions of virtually any type, on any subject, with accompanying answers. Use them to exercise your own memory or someone else's"

TIME (IN SECONDS) RESPONSE DISPLAY (0 TO 99) — This prompt is asking you to enter the length of time the *correct* answer will be displayed when a person enters a *wrong* answer. *It will not affect the length of time one has to enter an answer.* There is no time limit for a response.

After entering this information, the program will re-display your entries and ask you if they are all correct. If they *are* correct, then press Y. If you wish to change anything, press N. The program will then repeat the setup routine, asking you to re-enter all of the values.

Editing A Quiz

When you conclude the setup, you are taken to the Editor screen. The top of the screen displays the title and the current record number. The record number should be #1, the first record. Below this are two entry fields for the questions and answers. Above each field is the field prompt that you created on the setup screen. A sample question header for a quiz which teaches German might read GERMAN WORD, with the answer header ENGLISH WORD.

The cursor should be flashing inside the questions field. To enter a question, simply type it in. Completely filling the question field automatically transfers you to the answer field.

Enter GESUNDHEIT into the question field and, because it does not fill the question field, press (RETURN) or (ENTER) to terminate the input. After you enter the question, the cursor will move down to the answer field. The answer field is only one screen-line in size. Enter the answer GOOD HEALTH and press (RETURN) or (ENTER). The question and answer fields will clear, and the cursor will reappear in the question field for the next question. The record number at the top of the screen should now read #2.

This photo shows the IBM version administering a German language quiz. The Word Clues option displays five words at the bottom of the screen, one of which is correct.

Go ahead and experiment with your own questions and answers. When you are ready to save your quiz, press (RETURN) or (ENTER) when either the question or answer field is empty, and you will return to the Main Menu screen. If you accidentally do this before you've finished entering questions and answers, simply select 1) EDIT again to continue editing. If you exit from a blank answer field after entering a question, the question you entered for that record will be lost and you will need to re-enter it.

Searching And Changing Records

Back in the Edit mode, you can remodel the quiz you have built so far by searching for a string of characters in either the question field or the answer field. The keys used to select the search vary from

system to system, so check the Control Capsule for your machine. You can select either a question search, or an answer search. Once selected, the words SEARCH FOR will appear above the field. Enter the string that you want to search for in this field.

For example, if you want to locate the first question you entered, GESUNDHEIT, select the Question Search option, then enter GES. The program then searches for questions with the letter combination GES in them. You could have entered HEIT, or GESUND, and the search would have located the first record. When a record is found, its contents are displayed in the question and answer fields. If there is more than one record which matches the search characters or words you enter, then you can select the Next option from the choices listed below the question and answer fields:

PRESS C-CHANGE N-NEXT E-EXIT

If you press C, both the question and the answer fields will be cleared, and you will be able to re-enter them. Each time you press N, the program will continue to search for the next occurance of the search string you entered. You can keep searching—every record if necessary—until you find the record you want, or reach the end of the file. If you reach the end of the file and there are no other matches, the program will return to normal Edit mode, and the first blank record. This is also true when entering EDIT from the Main Menu.

If you wish to discontinue the search, press the key associated with EXIT. The actual key used to exit varies from system to system, so you will need to read your screen display or refer to the Control Capsule for your machine.

Save The Quiz File

To save your data, return to the Main Menu mode by pressing (ENTER) or (RETURN) in either field without entering anything else. Press 3 to select SAVE from the menu. The screen will clear, and then you will be asked to enter the following information:

QUIZ FILE NAME:
TODAY'S DATE:
YOUR NAME:

The QUIZ FILE NAME should be the file name you wish your quiz to have. On some systems you may be asked to also enter a device name or type of device, e.g., disk or tape. For TODAY'S DATE and YOUR NAME, you can enter anything you want to keep a history of the file. This information is displayed when the quiz is printed to the screen or a printer.

Once the save is complete, the program will report how many records were saved. To return to the Main Menu after saving, press (ENTER) or (RETURN).

"This software is actually two programs in one package . . .the first to construct and store your quizzes, and the second to retrieve and take them."

Load The Quiz File

Once you have created and saved a quiz to tape or disk, you may want to work on it again to expand it or make modifications. You can load the quiz by selecting option 2 from the Main Menu. You will be asked for the quiz file name. On some systems you may be asked to enter the device name. Enter the same file name used when you saved the quiz. The program will display information about the file as it's loaded:

title
LAST MODIFIED ON date
BY author's name
QUESTIONS: question header
ANSWERS: answer header

THERE ARE xx RECORDS

READING RECORD # xx

xx RECORDS LOADED
PRESS ANY KEY TO CONTINUE

Printing The Quiz

To list the quiz file contents for review, select option 4 from the Main Menu. You can list the quiz either to the screen, or to another device.

The information listed consists of the quiz parameters entered for the quiz on the parameter setup screen, followed by each question and answer in the quiz file.

Next issue we will present a third program, *Quiz-Print*, which will allow teachers to prepare hardcopy quizzes on a printer. With this program you will be able to format printed quizzes with a large number of options, as well as produce an answer sheet for grading purposes.

Change Parameters

If you have already created a quiz but would like to change its original parameters, select option 5. This will take you to the setup screen, and will ask you to re-enter all of the parameters. After entering them, you will be asked whether they are correct. If so, press Y and you will be returned to the Main Menu.

Exit The Program

If you have a quiz in memory and have made changes to it, then you will be notified before leaving the program, and be given an opportunity to return to the Main Menu. From the Main Menu, you can save the changed quiz, and then exit the program.

You can exit the program *without* saving the changes simply by indicating this when the exit routine warns you. If there have been no changes to the quiz, the exit routine will not stop you when you use option 6.

Quiz-Take

This program is used to take or study a quiz. You cannot alter the quiz from this program. If you wish to create or change a quiz, you must first use *Quiz-Make* to build or alter a quiz file.

After loading and running this program, you will be presented with a title screen. Press (ENTER) or (RETURN) to go to the Main Menu:

1) TAKE QUIZ
2) LOAD
3) STUDY QUIZ
4) EXIT

To select an option, simply press the number beside it. You do not need to press (RETURN) or (ENTER).

1) TAKE QUIZ — Before you can use this option to take the quiz, you will first need to use option 2 to LOAD a quiz.

2) LOAD QUIZ — This option must be used to LOAD a quiz file previously created with *Quiz-Make*. If you haven't yet created a file with *Quiz-Make*, then refer to the previous section on running that program. When you select this option you will be prompted to enter

a file name. On some systems you will be asked to also enter the device name—e.g., tape or disk, drive 1 or drive 2. The program will display the number of records read in from the file, and then wait for you to press (ENTER) or (RETURN) before continuing back to the Main Menu.

3) STUDY QUIZ — This option allows you to study a quiz. Four questions and answers will be displayed on the screen at a time. You can then scroll through the list of questions and answers by pressing the appropriate keys. (The keys used for each system are described in the Control Capsules included with this article.) You can exit this mode and return back to the Main Menu at any time by pressing the appropriate escape key, also described in the Control Capsules for each system.

4) EXIT — There are no restrictions in exiting this program as there are in *Quiz-Make*. You may exit the program at any time you like. You will never cause the loss of data by exiting the program because this program can not alter any files you have created. The only thing that may be lost by exiting the program is your score—and possibly your pride . . .

Quiz Level

After selecting the TAKE QUIZ option you will be shown another menu screen. This screen is used to select difficulty level of the quiz and the type of clues, if any.

1) WORD CLUES	2 TRIES
2) WORD CLUES	1 TRY
3) LETTER CLUES	3 TRIES
4) LETTER CLUES	2 TRIES
5) NO CLUES	1 TRY
6) SAME QUIZ	
7) EXIT	

1) & 2) Word Clues — If you select options 1 or 2, you will be given a list of five answers at the bottom of the screen for every question. One of those five answers will be the correct answer. The answers displayed are selected completely at random from all of the answers in the quiz file, so that each time the quiz is taken, or the same question is asked, there will be a different list of possible answers.

In option 1 you have two chances to answer a question correctly. If you miss the answer on the second try, a spelling check will be done to see whether you simply misspelled the word.

If you select option 2, you must answer each question on the first try. If an answer is incorrect, then the program will perform the spelling check.

3) & 4) Letter Clues — Options 3 and 4 will give you letter clues if you enter the wrong answer. The letter clues were explained in more detail in the *Quiz-Make* section "Percentage of Letter Clues."

In option 3, you are given three opportunities to answer a question. On the first try, no clues are given. If you miss the answer on the first try, then clues will be displayed in the answer field, with asterisks indicating character positions where a clue is not given. You can then type right over the clues and asterisks to enter your next answer.

If the second try is wrong, you will be given a new set of letter clues, and another chance at answering the question. If you miss the question on the third try, a spelling check will be performed.

If option 4 is selected, the quiz will act just as it did for option three, except that only *one* set of letter clues will be given. If you miss the answer on the second try, a spelling check will be performed.

5) No Clues — Option 5 will not give you any clues to the right answer, and will only allow *one* try to answer the question. If the answer is wrong on the first try, a spelling check is done to see if the answer has been misspelled.

6) Same Quiz — At any time during a quiz, you may return to the Main Menu screen by pressing (ENTER) or (RETURN). You may resume the quiz where you left off by selecting option 1 from the Main Menu (TAKE QUIZ), and then selecting option 6 from the quiz level menu (SAME QUIZ). The same quiz will be resumed with the score you had at the time you exited the quiz. If you select a quiz level other than option 6 (SAME QUIZ), the score will reset to zero and the quiz will start over.

7) Exit — Selecting option 7 will return you to the Main Menu.

Taking The Quiz

After selecting the quiz level, you will be taken to the quiz screen. This screen looks just like the one used for editing the quiz in the *Quiz-Make* program.

If the quiz is set up for sequential operation, then all of the questions have a predetermined order—they will be given in the same order in which they were entered. A question will be displayed in the question field, and the cursor will start blinking in the answer field, waiting for an answer to be entered. If a Word Clues option has been selected, it will be displayed at this time. After you enter the answer, it is checked against the correct answer. If it is not 100% correct, letter for letter, the answer is considered wrong. If the entry was the last try, or the *only* try (as in options 2 and 5), the answer undergoes a spelling check to see if the word is misspelled.

The spelling check is not 100% foolproof, but it does manage to catch minor spelling errors. The check is done by comparing each letter in your answer with the correct answer. Character position is important here. If 70% or more of the characters match, the answer is considered to be correct but misspelled. Less than a 70% match, and the answer is counted as being wrong. The comparison may look like this:

GOOD HEALTH	(Correct answer)
GOOD HAELTH	(9 out of 11— 81% — misspelled)
GOOD FOOD	(5 out of 11— 45%—wrong)

"At work or at play, the basic question and answer quiz is still a favored learning and teaching aid."

Notice that in the third answer above, *five* characters—not *four*—matched out of eleven because *all* of the characters are checked, even spaces. The alignment of the characters is important as well. If a character is simply left out, such as:

GOOD HEALTH	(Correct answer)
GOOD HALTH	(7 out of 11— 64%—wrong)

the characters to the right of the E in the word HELTH would not line up correctly with the characters in the correct answer, and would *all* be considered wrong. Thus, in this example only 64% of the characters match, making the answer incorrect—not just a misspelling.

Your score is displayed at the top of the screen during the quiz. The score is actually a percentage, and not a total. The percentages are for right answers, wrong answers, and misspellings. By putting the

misspellings in a separate category, placement of the score can be left up to the administrator of the quiz. It could be added to the right or the wrong answer score, or simply used as a separate method of evaluation. This "adding of the scores" must be done by the person giving the quiz—there are no provisions in the program to have it done automatically.

Administering Quizzes

To use these programs with a disk system, initialize two disks. Place *Quiz-Make* on one disk, and *Quiz-Take* on the other. Do all of your quiz development on the disk with *Quiz-Make*. When the quiz is complete, SAVE a copy of the quiz file to the disk with *Quiz-Take*. This will give you a back-up of the quiz file. In addition, the quiz taker will not be able to change the quiz file because the *Quiz-Take* disk does not contain the program *Quiz-Make*.

If you are using a cassette system, SAVE the two programs separately on two different tapes. Also, each quiz should be kept on its own tape. Label all tapes very clearly. This will prevent you from accidentally recording over one of the programs or the quiz file.

CONTROL CAPSULE
Quiz-Make

Edit Mode

KEY	FUNCTION
CTRL Q	Select question search mode.
CTRL A	Select answer search mode.
Left CRSR	Back space to erase.
RETURN	Return to menu.

Search Mode

KEY	FUNCTION
C	Change record.
N	Next record.
E	Return to menu.

Quiz-Take

Take Quiz Mode

KEY	FUNCTION
Left CRSR	Back Space to erase.
RETURN	Return to menu.

Study Quiz Mode

KEY	FUNCTION
Left CRSR	Scroll up.
Right	Scroll down
Esc	Return to menu.

Special Enhancement for Apple ProDOS

Under the Apple *II* family's new operating system, ProDOS, the system must be informed whenever you wish to access a disk with a different volume name or PREFIX. The program as published in the listing section was written to run under DOS 3.3, which does not use PREFIXes. To use the *Quiz Construction Set* programs when running under ProDOS, make the changes indicated in the following listings. [See the Apple-related "Home Computer Tech Note" in this issue for more information on ProDOS PREFIXes.—Ed.]

Make the following modifications to *Quiz-Make*:

```
 810 INVERSE:PRINT:"LOADER":DATA:NORMAL:
     PRINT:PRINT:PRINT:"ENTER FILENAME:
     ML=15:VT="HT=3:HT=17:GOSUB 1560
 820 FOR I=1 TO LEN(F$):A=ASC(MID$(F$,I,1)):
     IF A<64 OR A>90 THEN 240
 825 NEXT:T=1:GOTO 810
 830 IF NOT (V$="9" OR V$="Z") THEN PRINT CHR$(7):
     GOTO 810
 835 NEXT I
 836 PRINT:PRINT "INSERT DISK IN DRIVE 1 AND PRESS A KEY":
     GOSUB 1750:D$=CHR$(4)
 837 PRINT D$;"PREFIX,D1"
 980 PRINT:PRINT "ENTER FILE NAME:";:VT=9:HT=17:ML=15:GOSUB 1560
 990 FOR I=1 TO LEN(F$):A=ASC(MID$(F$,I,1)):
     IF A<64 OR A>90 THEN 240
 995 NEXT:T=1:GOTO 980
1000 IF NOT ((V$="." OR V$="9") OR (V$>"Z" OR V$<"A")) THEN PRINT CHR$(7):
     GOTO 810
1005 NEXT
1006 PRINT:PRINT "INSERT DRIVE 1 AND PRESS A KEY":
     GOSUB 1750:D$=CHR$(4)
1007 PRINT D$;"PREFIX,D1"
1010 PRINT D$;"OPEN";F$
```

Make the following modifications to *Quiz-Take*:

```
 800 PRINT:PRINT "ENTER FILE NAME:";:VT=17:GOSUB 1020:
 810 IF B$="" THEN 230
 815 FOR I=1 TO LEN(F$):IF MID$(F$,I,1)<"0" OR MID$(F$,I,1)>"9" THEN 240
 820 IF NOT ((V$="." OR V$="9") OR (V$>"Z" OR V$<"A")) THEN PRINT CHR$(7):
     GOTO 790
 825 NEXT:I=1
 826 PRINT:PRINT "INSERT DISK IN DRIVE 1 AND PRESS A KEY":
     GOSUB 1020:D$=CHR$(4)
 827 PRINT D$;"PREFIX,D1"
 830 PRINT D$;"VERIFY";F$
 840 PRINT D$;"OPEN";F$
```

CONTROL CAPSULE
Quiz-Make

Edit Mode

KEY	FUNCTION
SHFT Q	Select question search mode.
SHFT A	Select answer search mode.
DEL	Back space to erase.
RETURN	Return to menu.

Search Mode

KEY	FUNCTION
C	Change record.
N	Next record.
E	Return to menu.

Quiz-Take

Take Quiz Mode

KEY	FUNCTION
SHFT X	Return to menu.
DEL	Back space to erase.

Study Quiz Mode

KEY	FUNCTION
Down CRSR	Scroll down.
Up CRSR	Scroll up.
SHFT X	Return to menu.

CONTROL CAPSULE
Quiz-Make

Edit Mode

KEY	FUNCTION
F1	Select question search mode.
F2	Select answer search mode.
Back Space	Back space to erase.
ENTER	Return to menu.

Search Mode

KEY	FUNCTION
C	Change record.
N	Next record.
ENTER	Return to Edit mode.

Quiz-Take

Take Quiz Mode

KEY	FUNCTION
ENTER	Return to menu.
Back Space	Back space to erase.

Study Quiz Mode

KEY	FUNCTION
F1	Scroll up.
F2	Scroll down.
Esc	Return to menu.

CONTROL CAPSULE
Quiz-Make

Edit Mode

KEY	FUNCTION
FCTN E	Select questions search mode.
FCTN X	Select answers search mode.
FCTN 3	Back space to erase.
ENTER	Return to menu.

Search Mode

KEY	FUNCTION
C	Change a record.
N	Next record.
E	Return to Edit mode.

Quiz-Take

Take Quiz Mode

KEY	FUNCTION
FCTN 9	Return to menu.
FCTN 3	Back space to erase.

Study Quiz Mode

KEY	FUNCTION
FCTN 9	Return to menu.
FCTN E	Scroll up.
FCTN X	Scroll down.

SAMPLE DATA BASES

THREE QUIZZES READY-TO-RUN

Here are three quizzes that you can enter into the *Quiz Construction Set* right away. Each of the first two quizzes is divided into 2 parts: Part 1 of the English vocabulary quiz places words in each answer field, and their definitions in the question fields. Part 2 features words used only in a special context, such as trade jargon or slang words—with each word given as a question, and each definition as an answer. Part 1 of the German quiz lists common German words in the question fields, and their meanings in the answer fields. In part 2, each question is a common English phrase and each answer, the German equivalent. (We suggest you use a Word Clues option for part 2 of each quiz.) A short trivia quiz is the last sample data base.

The questions and answers given here can be used with all four versions of the program. You can also add to these quizzes, use only those questions and answers you want, or create a quiz comprised of questions from all of the examples shown here.

VOCABULARY QUIZ 1

Q. harsh or discordant sound
A. cacophony

Q. decaying organic matter found on the forest floor
A. duff

Q. the space between nerve cells
A. synapse

Q. term for a fertilized egg
A. zygote

Q. a riddle
A. conundrum

Q. act of giving birth
A. parturition

Q. calm, happy, and peaceful
A. halcyon

Q. fear of foreigners
A. xenophobia

Q. a sudden outburst
A. salvo

Q. an article of food
A. viand

Q. to talk informally; chat
A. confabulate

Q. tending to melt or dissolve
A. deliquescent

Q. resembling a tree in structure
A. dendriform

Q. irritable or peevishly sensitive
A. tetchy

Q. inappropriately jocular
A. facetious

Q. having no petals
A. apetalous

Q. to cheat out of what is due
A. bilk

Q. funnel-shaped clay smoking pipe
A. chillum

Q. extremely cold
A. gelid

Q. following in time or order
A. subsequent

Q. just and fair; impartial
A. equitable

Q. social courtesies; manners
A. amenities

Q. characterized by verbal abuse
A. vituperative

Q. liberating energy
A. exergonic

VOCABULARY QUIZ 2

Q. saute'
A. fry in small amount of fat

Q. schuss
A. ski downhill at high speed

Q. allegro
A. a fast tempo in music

Q. gaffer
A. movie lighting technician

Q. plumb
A. straight up and down

Q. tweek
A. to adjust (electronics)

Q. byte
A. eight bits of data

Q. bullish
A. optimistic of boom market

Q. overdub
A. to record sound on sound

Q. codex
A. a manuscript book

Q. build-down
A. keep only new weapons

Q. totem
A. emblem or revered symbol

Q. gable
A. end wall of a building

Q. chutzpah
A. extreme self-confidence

Q. frappe
A. a partly frozen drink

Q. piquant
A. engagingly provocative

Q. pixel
A. screen picture element

Q. bug
A. a program malfunction

Q. hyperbole
A. extravagant exaggeration

Q. sprent
A. sprinkled over

Q. yarder
A. a log-pulling machine

Q. vapid
A. lacking liveliness

Q. gaggle
A. flock of geese on ground

Q. parry
A. to ward off an attack

Q. maquette
A. small preliminary model

Q. perquisite
A. extra reward or gratuity

Q. farrier
A. one that shoes horses

Q. warp
A. lengthwise strings in loom

GERMAN QUIZ 1

Q. der Koffer
A. suitcase

Q. gutaussehend
A. good-looking

Q. die Reise
A. trip

Q. nuelich
A. recently

Q. zwischen
A. between

Q. augenblichlich
A. immediately

Q. die Brieftasche
A. wallet

Q. eingebildet
A. egotistical

Q. das Fliessband
A. assembly line

Q. schlafen
A. to sleep

Q. die Innenstadt
A. downtown

Q. wiederholen
A. to repeat

Q. vergebens
A. in vain

Q. die Gemeinschaftschule
A. primary school

Q. die Schreibwaren
A. stationery

Q. zeigen
A. to show

Q. das Verfahren
A. procedure

Q. furchtbar
A. horrible

Q. wunderlich
A. wonderful

Q. die Armbanduhr
A. wristwatch

Q. ungezwungen
A. casual

Q. die Verwandten
A. relatives

Q. zugeben
A. to forgive

Q. das Rasiermesser
A. razor

Q. schlank
A. slender

Q. jawohl
A. indeed

Q. verstehen
A. to understand

GERMAN QUIZ 2

Q. in the meantime
A. in der Zwischenzeit

Q. what's the matter?
A. was ist los?

Q. take care!
A. mach's gut!

Q. I am sorry
A. es tut mir leid

Q. it's now or never
A. entweder jetzt oder nie

Q. it works wonders
A. wunder wirken

Q. do you have a light?
A. haben sie Feuer?

Q. help yourself
A. sich bedienen

Q. now and then
A. hin und wieder

Q. for example
A. zum Beispiel

Q. in that case
A. in diesem Fall

Q. take it easy
A. nimm's leicht

Q. without a doubt
A. ohne Zweifel

Q. be that as it may
A. wie dem auch sei

Q. hurry up
A. mach schnell

Q. in the morning
A. am Morgen

Q. how are you?
A. wie geht's?

Q. good day!
A. guten Tag!

Q. we have a lot in common
A. sie steht mir nahe

Q. you're welcome
A. bitte sehr

Q. show me
A. zeigen sie mir

Q. what does that come to?
A. wieviel macht das?

Q. that turns me on
A. das begeisert mich

Q. a big shot
A. ein hohes Tier

Q. time is up
A. die Zeit ist um

Q. see you later!
A. Aufwiedersehn!

TRIVIA QUIZ

Q. Who was the fifth Marx brother?
A. Gummo

Q. What's the flip side of the Beatles' single, "Rain"?
A. Paperback Writer

Q. To what religious sect do we owe the circular saw?
A. Shakers

Q. Which was the 1st major car with front wheel drive?
A. The Cord

Q. Who is the robot in "The Day the Earth Stood Still"?
A. Tobar

Q. Who are the people of "The Forbidden Planet"?
A. The Krell

Q. What TV show featured Cochise?
A. Broken Arrow

Q. Name the first computer to use a mouse and icons.
A. Xerox Star

Q. Who shot James A. Garfield?
A. Charles Guiteau

Q. How many typographic points to the inch?
A. 72

Q. Which particle has both light and matter properties?
A. nuetrino

Q. Which famous cowboy movie star carried a whip?
A. Lash La Rue

Q. What is the world's highest-flying jet aircraft?
A. SR-71 Blackbird

Q. Who was Fred Flintstone's best friend?
A. Barney Rubble

Q. Where was the first semiconductor produced?
A. Bell Labs

Q. Who is the father of the Pascal language?
A. Nicholas Wirth

Q. What is Ringo's other name on the Sgt. Pepper album?
A. Billy Shears

Q. Who said, "I have not yet begun to fight"?
A. John Paul Jones

Q. In which film did Chaplin satirize Adolf Hitler?
A. The Great Dictator

Q. Who was the "Man of a Thousand Faces"?
A. Lon Chaney Sr.

Q. What was the name of the dog in "Topper"?
A. Neil

Q. What is the name of Ricky's brother in "Ozzie and Harriet"?
A. David

Q. What is the largest self-supporting concrete roof?
A. The Seattle Kingdome

Q. Who was the founder of the Republic of China?
A. Sun Yat-sen

Q. Which computer magazine has no outside advertising?
A. Home Computer Magazine

Q. Who said, "I will fight no more forever"?
A. Chief Joseph

Q. What is the name of the spice in "Dune"?
A. melange

Q. Who hosted "You Are There" in the 1950's?
A. Walter Cronkite

Q. What were Romeo and Juliet's last names?
A. Montegue and Capulet

Q. Which President was raised as a Quaker?
A. Richard Nixon

Q. What was Priam's prize for judging a beauty contest?
A. Helen

Q. Which early radio show restaged movie hits?
A. Lux Radio Theater

Q. What was the dry planet in "The Dispossessed"?
A. Anarres

Q. Who was the housekeeper in "My Three Sons"?
A. Bub

Q. What substance powers the Starship Enterprise?
A. dilithium crystals

Quiz-Make (Apple *II* Family) Explanation of the Program

Line Nos.	
100-180	Program header.
190	Define error-trapping routines location.
200-230	Title screen.
240-290	Main menu.
300-800	Edit quiz and search records.
810-940	Load quiz file.
950-1100	Save quiz file.
1110-1320	Print quiz file.
1330-1470	Input-parameters routine.
1480-1540	Exit-program routine.
1550	Illegal entry message.
1560-1740	Main-input routine.
1750-1760	Single-key-input routine.
1770-1810	Error-trapping routine.
1820-1830	Program data.

Quiz-Make (C-64) Explanation of the Program

Line Nos.	
100-170	Program header.
180-250	Display title screen.
260-370	Main menu.
380-750	Edit the quiz.
760-1130	Search mode.
1140-1450	Load the quiz file.
1460-1780	Save the quiz file.
1790-2200	Print the quiz file.
2210-2510	Change-parameters routine.
2520-2630	Input routine.
2640-2720	Illegal entry messages.
2730-2870	Input-a-question routine.
2880-3020	Input-an-answer routine.
3030-3040	Clear parts of the edit screen.
3050-3060	Routine to locate the cursor.
3070-3170	Exit-program routine.

Quiz-Make (IBM PC, PCjr)
Explanation of the Program

Line Nos.	
100-200	Program header.
210	Define error-trapping-routines location.
220-250	Initialization and title screen.
260-300	Main menu.
310-630	Edit and search mode.
640-740	Load quiz file.
750-850	Save quiz file.
860-1010	Print quiz file contents.
1020-1060	Controlling routine for change parameter option.
1070-1100	Single-key-input routine.
1110-1220	Main-input routine.
1230-1290	Error-trapping routine.
1300-1350	Program data.
1360-1420	End-of-program routine.

Quiz-Take (Apple II Family)
Explanation of the Program

Line Nos.	
100-180	Program header.
190	Define error-trapping routines location.
200-220	Title screen.
230-280	Main menu.
290-390	Quiz level menu.
400-420	Display the quiz screen.
430-520	Display problem and get answer.
530-600	Branch to appropriate routine depending on the option selected from the quiz option menu.
610-630	Wrong answer.
640-660	Display letter clues.
670-750	Spelling check.
760-770	Wrong answer—try again.
780	Right answer.
790-910	Load a quiz file.
920-1010	Study the quiz mode.
1020-1190	Main-input routine.
1200-1210	Clear parts of the screen.
1220-1260	Select five random word clues.
1270-1290	Display scores.
1300-1320	Single-key-input routine.
1330-1370	Error-trapping routine.
1380-1390	Exit-program routine.
1400-1410	Program data.

Quiz-Take (IBM PC, PCjr)
Explanation of the Program

Line Nos.	
100-200	Program header.
210	Define error-trapping-routines location.
220-250	Initialization and title screen.
260-300	Main menu.
310-390	Quiz options menu.
400-470	Display the quiz screen.
480-490	Display question, get answer, and branch to routine to handle response depending on quiz level selected.
500-540	Do spelling check.
550-570	Display letter clues.
580	Correct answer.
590-660	Study mode.
670-760	Load the quiz file.
770-800	Single-key-input routine.
810-900	Main-input routine.
910-980	Error routine.
990-1020	Time-delay routine.
1030-1060	Program data.
1070-1100	End-of-program routine.
1110-1140	Display-scores routine.

Quiz-Make (TI-99/4A)
TI BASIC or Extended BASIC
Explanation of the Program

Line Nos.	
100-170	Program header.
180-240	Initialization and title screen.
250-310	Main menu.
320-1380	Edit quiz and search records.
1390-1530	Load a quiz file.
1540-1720	Save a quiz file.
1730-2080	Print a quiz file.
2090-2710	Change parameters.
2720-3080	Main-input routine.
3090-3130	Illegal entry message.
3140-3170	Single-key-input routine.
3180-3270	Routines to clear the question and answer fields.
3280-3340	Routine to clear parts of the edit screen.
3350-3440	Exit-program routine.

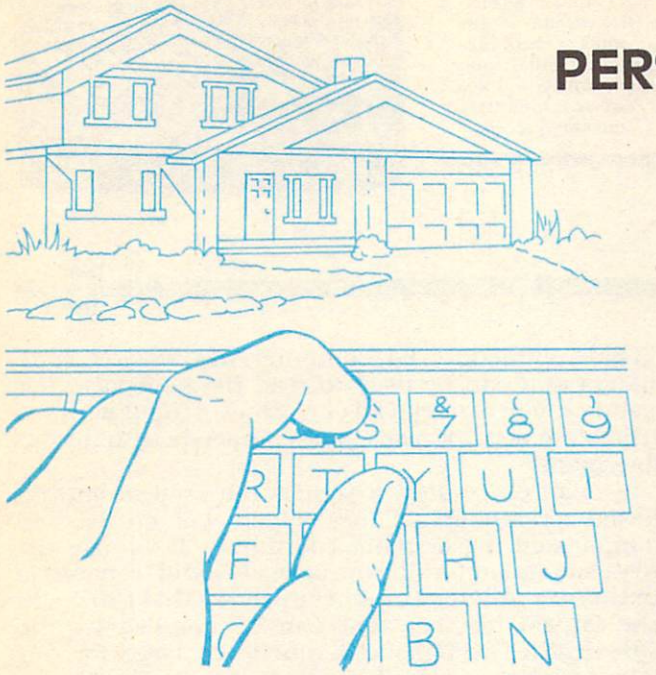
Quiz-Take (C-64)
Explanation of the Program

Line Nos.	
100-170	Program header.
180-250	Display the title screen.
260-380	Main menu.
390-550	Option menu for level of difficulty.
560-700	Display the quiz screen.
710-790	Display question and get answer.
800-880	Branch to routine to handle user's response.
890-960	Wrong-answer routine.
970-1070	Display letter clues.
1080-1200	Check spelling.
1210-1250	Missed guess—try again.
1260-1420	Right answer.
1430-1740	Load a quiz file.
1750-1870	Study-quiz routine.
1880-2000	Display the scores.
2010-2110	Choose five random numbers.
2120-2180	Clear the question and answer fields.
2190-2200	Locate-cursor routine.
2210-2330	Input routine.
2340-2360	Single-key-input routine.
2370-2420	Illegal entry messages.
2430-2450	End-of-program routine.

Quiz-Take (TI-99/4A)
TI BASIC or Extended BASIC
Explanation of the Program

Line Nos.	
100-180	Program header.
190-250	Title screen.
260-370	Main menu
380-560	Quiz options menu.
570-900	Display the quiz screen.
910-1030	Display the problem and get the answer.
1040-1130	Branch to appropriate routine depending on the level selected from the quiz level screen.
1140-1290	Wrong-answer routine.
1300-1410	Display letter clues.
1420-1690	Spelling check.
1700-1740	Wrong answer—get another try.
1750-1840	Right answer routine.
1850-2080	Study quiz mode.
2090-2230	Display scores.
2240-2350	Select five random word clues.
2360-2390	Clear portions of the quiz screen.
2400-2440	Routine to display messages on the screen.
2450-2770	Main-input routine.
2780-2800	Single-key-input routine.
2810-2820	Exit-program routine.

HCM



PERSONAL LOAN CALCULATOR

by H. W. Button
and the HCM Staff

Borrowing money? Here's a program to give you all the information you need to make that loan pay off.

Personal Loan Calculator is a handy companion to the *Savings* program published in Vol. 4, No. 1 for the Apple, Commodore and IBM computers, and in Vol. 2, No. 6 for the the TI-99/4A. The two programs together form a comprehensive software package for everyday personal financial decisions.

Borrowing money can be terrifying. A long-term loan, such as a home mortgage, may be considered a brave act—and certainly sometimes a confusing one.

Where Your Interest Lies . . .

Many inexperienced borrowers are surprised to find that for the first few months, or even years, of their payment schedule, they will be paying primarily the *interest* on their loan. Only after most of this interest is covered will they begin to pay off a significant portion of the *principal*—that is, the actual amount of money borrowed. With each payment, the proportion of interest to principal changes: the percentage of the payment going to interest *decreases*, and the percentage going to the principal *increases*.

Many questions arise when someone considers either an existing or a prospective loan: How much will I end up paying over the entire term of the loan? How many payments will I have to make? How big will the payments be? How big a loan can I afford? At any given time, how much will I be paying to interest, and to principal? And how much principal will I have left to pay? The *Loan Calculator* is designed to answer these questions with the use of your home computer.

Displayed on the menu of *Loan Calculator* are five options:

1. Payment amount
2. Number of payments
3. Loan amount
   (How much you can afford to borrow.)
4. Amortization Schedule
5. Exit the program

You may select any one of these options, in any order. In options 1, 2, or 3, you are first asked two questions: The first inquires whether your payments will be made monthly or annually. The second asks whether the length of the loan period should be expressed in months or years. This input will affect how often the interest is compounded. Compounding the interest will alter the amount of interest due for each payment, according to the amount of principal still owed on the loan. For example, an annual interest rate of 12% will compound monthly at 1%. If you had $10,000.00 in principal left to repay, the interest for one month would be 1% of $10,000, or $100.

In Option 1, you also will be asked the following:
- Interest rate?
- Months (years) of loan?
- Amount of loan?

In Option 2 you will be asked:
- Interest rate?
- Monthly (annual) payment?
- Amount of loan?

If the payment amount you enter in Option 2 is too low to cover the interest generated during each payment period, you will receive an error message asking you to enter new data (either raise your payment or lower the interest rate or loan amount).

In Option 3 you will be asked:
- Interest rate?
- Monthly (annual) payment?
- Months (years) of loan?

"Many inexperienced borrowers are surprised to find that for the first few months, or even years, of their payment schedule, they will be paying primarily the interest on their loan."

Using any of these first three options will generate a report which consists of the following new information, plus the data you have already entered:

Interest rate	Compounded (monthly or annually)
Loan amount	Payment amount
Number of payments	Term of the loan
Total cost (principal + interest)	Total interest

In this report (see photo), the program will supply the data you have *not* entered, but want to know.

Finally, in Option 4 (Amortization Schedule), you will be asked the following:

Loan amount?
Number of monthly payments?
Interest rate?

After you enter this data, the following information will be printed:

Monthly payment
Final payment

You will then be prompted for the following:

Show schedule from payment #?
To payment #?

Here you enter the starting and ending payments that you want included in the report. The report for all of these months will then be displayed, one month at a time. This report will include:

Payment #
Interest for this payment
Principal for this payment
Loan balance for this payment

You may scroll through the payment schedule month by month, and return to the Main Menu after viewing the last monthly report.

This *Loan Calculator* program is a very handy, flexible tool for quickly generating information vital to anyone who is either considering or already paying off a loan. If you are in one of these positions, this program is "just what the banker ordered."

For *your* key-in listing see HCM PROGRAM LISTINGS Contents on page 85.

[**NOTE:** When using *Loan Calculator* on the IBM PC, make sure to start the BASIC language by typing in BASICA/D to enable Double Precision mode—Ed.]

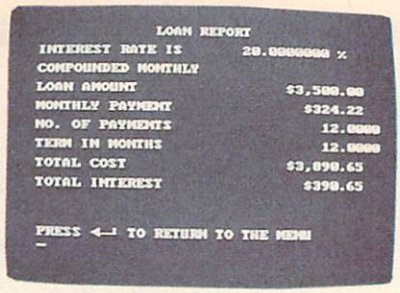
The IBM PC and PCjr have a very powerful function built into their BASIC PRINT statement which allows display formatting. The USING option of the PRINT statement allows a programmer to specify certain formatting parameters for the display. The most common type of formatting is done with numeric values. Three format types are used in this program, as follows:

```
F$      $$###,###,###,###.##
IST$    ########.######## %
F2$     ##################.####
```

The first format, F$, is used to display all monetary values. The double dollar sign ($$) indicates that a floating dollar sign will be displayed to the left of a number, and each pound sign represents a digit. If there aren't enough digits to fill all of the pound signs, they will be replaced with spaces. The commas will be printed only if there are enough digits in the number to fill the pound signs to the left of the comma. The two pound signs to the right of the decimal point indicate that the value will be rounded off to two decimal places. If the value is an integer, or has only one decimal place, the pound signs to the right of the decimal point will be replaced with zeros (0). The number 2483.1 would be displayed as: $2,483.10.

The second format shown above is used to display the interest rate. The third format is used for other miscellaneous numeric displays.

As shown in this IBM version of the report screen, the program will display the information you entered, plus the missing pieces.

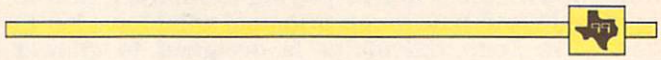


The Commodore 64 computer has several commands that can be used to read the keyboard. For instance, you can read an entire line of input with the INPUT statement, or read single characters with the GET statement.

In *Loan Calculator*, a user is required to enter a variety of numeric values which will be used in complicated mathematical formulas. If we use the INPUT statement for this purpose, it would be possible for the user to enter a number up to 1.70141183E+38 (the largest number that can be handled by the C-64 BASIC system, and much too large for our calculations). And, if a user enters alphabetic characters instead of numbers, a lot of limit checking would be required after an entry is made. Any errors would mean that the entire entry would have to be done over again.

However, there is a solution—the GET statement can read one character at a time. If we include a limit check on each character as it is typed, we can eliminate any illegal characters before they can be added to the user's input. This is done in a subroutine (lines 1700-2030) that is called every time an input is required. Thus, the program can pass several variables to the routine to prevent illegal entries. Below is a list of the variables that are passed to the routine. This routine may come in handy in some of your own programming ventures.

Variable	Function
LN	Screen line where input will appear
HV	High value limit check
LV	Low value limit check



[**Special Note:** Due to space limitations, we are publishing only an Extended BASIC version of *Loan Calculator* in this issue of *HCM*. Extended BASIC provides for a more elegant solution to many problems, due to its superior ability to handle error-checking, and its advanced input and display functions. Recognizing, however, that *Loan Calculator* is a highly useful program, and that many TI-99/4A owners do not have Extended BASIC, we have also written a TI BASIC version that can be found along with the Extended BASIC version on this issue's (Vol. 4, No. 5) TI edition of ON TAPE or ON DISK. See back cover and order card for information.]

This Extended BASIC version of *Loan Calculator* takes advantage of the USING option of the DISPLAY AT statement to format the values displayed on the report screen. To indicate the format to be used with the USING option, you must specify the line number that contains the IMAGE statement. The IMAGE statement must be the only statement on that line; it contains a string of characters which indicate the output format. The three formats used in this program are on line numbers 1840, 1850, and 1860. To use the format in line 1840, place USING 1840 in the DISPLAY AT statement as is done in line 1440.

The first two statements in the program will trap any errors that may occur in the program. There are two levels to the error-trapping. If the error is serious enough, it will be trapped by the ON ERROR statement, and the program will branch to an error-handling routine which starts in line 1870. The error routine will flash a message on the screen telling the user that there was an error in the calculation, meaning that the program can't finish it. The program will then return to the Main Menu screen. Minor errors will simply be passed over because of the ON WARNING NEXT statement.

Because the Apple computer has no formal way to format numeric data displayed on the screen, it is necessary to do it manually within the program by employing a rounding algorithm.

For example, when you display dollars and cents, you obviously don't want the value to contain more than two digits to the right of the decimal point—i.e., $123.456 isn't correct. The value $123.46 is what you are seeking. It is possible to achieve this result with a simple rounding algorithm. If the value you want to round off is contained in the variable A, then the following line will round it to the nearest cent:

```
A=INT(100*A+.5)/100
```

This works by first multiplying A by 100: 123.456 would become 12345.6 in value. The program would then add .5 to that value, giving us 12346.1 as an intermediate result. The INT function would then chop off the .1, returning only the integer part of the number 12346. Finally, this result is divided by 100, giving us 123.46 as the final displayed result. If you desire a result that is not rounded, but is still limited to only two digits past the decimal point, then omit the +.5 from the equation.

Loan Calculator (Apple *II* Family) Explanation of the Program

Line Nos.	
100-170	Program header.
180-230	Main menu.
240	Exit program.
250-290	Enter frequency of payments, and the expression for the length of the loan period.
300-490	Option 1—Payment amount.
500-660	Option 3—Amount of loan.
670-810	Option 2—Number of payments.
820-930	Final report for options 1, 2, and 3.
940-1160	Option 4—Amortization Schedule.
1170-1220	Routine to format the display output.

Loan Calculator (TI-99/4A) Explanation of the Program

Line Nos.	
100-170	Program header.
180-200	Title screen.
210-280	Main Menu screen.
290-370	Enter values for frequency of payments, and the expression used for the length of the loan period.
380-690	Option 1—Amount of payment.
700-1020	Option 2—Total loan amount.
1030-1380	Option 3—Number of payments.
1390-1530	Display the final report for options 1, 2, and 3.
1540	End the program.
1550-1830	Option 4—calculate and display the Amortization Schedule.
1840-1860	IMAGE formats for the USING option.
1870-1880	Error routine.

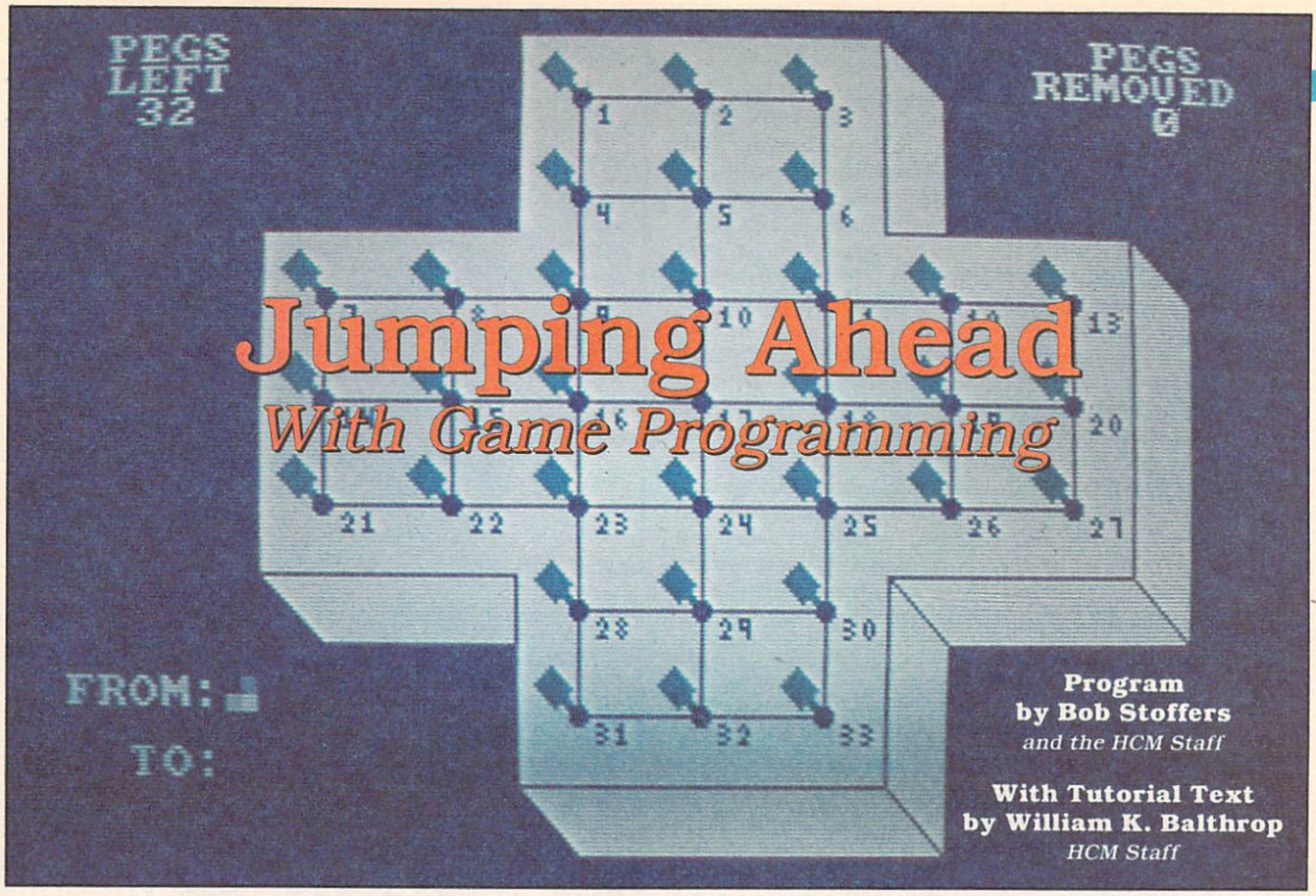
Loan Calculator (IBM PC, PCjr) Explanation of the Program

Line Nos.	
100-190	Program header.
200-230	Initialize program.
240-290	Main Menu screen.
300-350	Enter frequency of payments, and the expression of the term of the loan.
360-440	Subroutines for all inputs.
450-540	Option 1—Amount of payments.
550-640	Option 2—Number of payments.
650-740	Option 3—Amount of loan.
750-870	Final report for options 1, 2, and 3.
880-1090	Option 4—Amortization Schedule.
1100-1210	Utility subroutines.
1220-1230	Error-handling routine.
1240-1250	(Esc)-key routine.
1260-1270	Exit-program routine.

Loan Calculator (C-64) Explanation of the Program

Line Nos.	
100-180	Program header.
190-300	Main Menu.
310-420	Get inputs for length of loan period and term expression.
430-630	Option 1—Routine for amount of payment.
640-860	Option 2—Routine for total loan amount.
870-1110	Option 3—Routine for number of payments.
1120-1360	Display final report for options 1, 2, and 3.
1370-1850	Option 4—Routine to calculate and display the Amortization Schedule.
1860	Exit program.
1870-1890	Error-in-calculations routine.
1900-2030	Input routine. Checks that all entries are numeric, and within the specified value ranges.

HCM



There's more to home computing than just playing games—You can create the games as well. Learn how by following along as we analyze this video version of the classic peg-jumping puzzle.

[**Note:** In Vol. 4, No. 3 of *HCM*, an article entitled, "Programming: The Name of the Game" presented some helpful tips on game programming, looking at specific coding and trouble-shooting techniques. In the article below, we look at the structure of one particular game, and how it reflects important principles of game design.]

Have you ever wanted to produce your own video games? Perhaps—even though you may be so-inspired—you have shied away from the actual task of programming an original game, and you've been content to play whatever comes along. Still, some of the most fun you can have with your home computer is not just *playing* games, but *creating* them.

We would like to encourage you to try your own hand in game programming by illustrating how a game is first designed and then transformed to program code. In the process—and as an example—we will introduce *Peg Jump*, a board game that will challenge your puzzle-solving skills for many hours.

Getting Started

Sometimes, the hardest part of writing a game program is coming up with an idea for it. If this is a problem for you, try sitting back in a comfortable chair with your eyes closed, and daydream a little. Think about what you would like your computer to be able to do. At this point, don't worry about how such a program might work, think only about what it would do. Thinking about details at this time will probably cloud the imaginative process.

One important point, however—don't try to simply copy someone else's program or idea. This exhibits the lowest level of imagination, and, if carried too far, may infringe on copyright laws. Other programmers' ideas can show you what is *possible*, inspiring you to create your *own* original (and practical) ideas.

In *Peg Jump* the idea was to make a game which simulates a puzzle that many of you may have played as children. The puzzle is known by several names, and consists of a playing board in the shape of a cross, with 33 holes in it. The game starts with a peg in every hole except the center. The object is to jump one peg at a time over another peg, removing each peg jumped as you go. To win, you must finish the game with just one peg left on the board, in the center hole.

Most computer games fall into one of the following categories: Board game, Puzzle, Arcade, Adventure, or Education.

Any game may actually cross over into two or more of these categories. *Peg Jump* is a combination board game and puzzle. It's a board game because the action takes place on a playing surface which could just as easily be a game board on your kitchen table. For example, Chess and Backgammon—two computer favorites—are both board games. *Peg Jump* is also a puzzle because it challenges the player to solve a problem through a series of moves—in a sense, playing *against* the board.

Preliminary Design

At this stage you should already know what you want your program to do. Now you need to sit down and create a general outline of the events that will take place in the program. This will help you organize your thoughts, and result in a more complete design on the first pass.

Your general outline may take the shape of this example used for *Peg Jump* :

1. Initialize the game.
2. Draw a board on which to play the game.
3. Let the player enter a move.
4. Check for legal moves.
5. Move the peg from one hole to the next; delete the peg that was jumped.
6. Check for an end-of-game situation.
7. Have the puzzle solve itself upon request.

If you work from a list like this, and tackle one problem at a time, you will usually end up with a much cleaner program.

Another preliminary step—creating a flow chart—is optional, but recommended. Often, programmers create a flow chart of their program before they write its code. A flow chart is a series of boxes connected by lines; each box contains a function or decision that the program performs. The lines connecting the boxes show the "flow" of the program. By creating such a chart you can figure out what kind of routines the program will need.

Other programmers find a flow chart too restrictive and prefer to work from general outlines (like the one above), structured charts, or even scraps of paper. The important thing is that you, the programmer, have a clear idea of all aspects of the programming task, and a carefully-laid plan of attack to accomplish it before you start coding.

Starting to Write the Code

Now you're ready to sit down at your computer and start banging away at the keyboard. Don't forget to make a large pot of hot coffee, and say goodbye to your family and friends. Then GOTO it!

Writing the code isn't as scary as it may seem at first. A good place to start is with a title screen. You can even get fancy if you want, making large letters or adding graphics to spruce it up. Keep one point in mind here—do not leave players hanging without letting them know what to do. An example of this would be a program which displays a title screen, and then sits there waiting for the player to press a key to continue—without any message on the screen. This may seem trivial, but you would be surprised at how many programs don't give adequate screen instructions. On the other hand, this shouldn't mean 40 pages of screen text either. Keep all screen messages as short and informative as possible.

Initialize The Program

After you have designed the title screen, you should implement the initialization phase of the program. This is the part of the routine that sets up everything for the rest of the game. It also should ensure that the program is going to run in the proper environment (memory size, graphics options, etc.).

Ask yourself a few questions before starting this section. Will you need to use an array, and if so, how big does it need to be? Do you need to create any graphics patterns, or assign values to key variables? You may need to come back and add to this section from time to time as the program develops, and you get a better feel for what is needed.

If the program must go through a long initialization with nothing happening on screen, you should make users aware of the delay; otherwise they may think the program or computer isn't working properly. This is true on the Texas Instruments, Commodore, and Apple versions of *Peg Jump*—they all go through a fairly lengthy initialization, and so display the message PLEASE STAND BY.

"Ideally, a game should be so easy to operate that a player can begin without even reading the instructions—no matter how complex the game actually may be."

Draw a Game Board

This part of the program sets up the playing screen for the rest of the game. Board games are generally the easiest type of screens to set up because most of them are static (remain the same). Of primary importance to the enjoyment of your program is appearance. An attractive and exciting screen will keep people interested in the game longer than if the screen were dull and boring. This is especially true for board games and puzzles, where the action is usually very slow.

Before you start placing graphics randomly on the screen, you should first sketch them out on a piece of graph paper with a pencil and a good eraser. You may save yourself hours of programming headaches by first proving that a screen design works on paper. You can then use the graph-paper grids to help calculate screen coordinates. This is really an invaluable step in the design process, without which you may spend countless hours designing and re-designing your display, only to finally end up with a lackluster screen.

In *Peg Jump* , we need to place an imaginary board in the shape of a cross on the screen. We must somehow illustrate the 33 holes in the board, and indicate which holes have pegs in them. The actual method used varies from system to system, so we won't go into great detail on the written code. It is easy to draw the board on a piece of paper, and then locate where the holes should line up. From this, it is possible to calculate where on the screen each hole is located. Once this is known, the coordinates can be stored in an array. To find the location of any given hole, you simply need to index into the array, and locate the hole's screen coordinates.

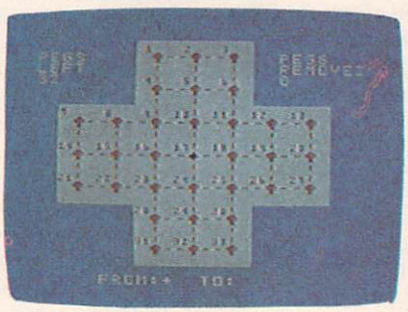
But how will the player know which holes are which? Remember our discussion about giving the player adequate information? If a player had to sit there and count all the holes to find the desired hole number in order to make a move, the game would soon become more of a headache than a pleasure. The best way to handle the situation is to number each hole with its index value in the array that contains the hole's location. This will also simplify the program code. We use small numbers next to each hole so that we don't clutter the board with large, awkward figures.

Knowing the Score

The final touch is a display of the player's score. In this case the display indicates how many pegs are left on the board, and how many pegs have been removed. The score display should be placed on the screen where the player can glance at it without losing concentration; it should be available, but not a distraction. Depending on the situation, you may choose not to display the score until the end of the game, although this technique is not used very often. Most people, when playing a game, are inspired by trying to increase their score. If they can't see their score during the game, they have no incentive to try to improve.

Player Interaction

This part of a program is the interface between the player and the computer. It is very critical, and should be given close attention. If a player must make awkward moves, if the commands are difficult to remember, or if the keys to control the game don't make any sense, the game loses a lot of appeal—and in arcade-type games, a lot of realism. Ideally, a game should be so easy to operate that a player can begin without even reading the instructions—no matter how complex the game actually may be. If necessary, provide brief screen instructions to indicate what type of action is required of the player. If the computer is waiting for the player to enter a number, as in *Peg Jump*, let the player know what you want. In *Peg Jump*, the message FROM: lets the player know that the computer is waiting for a number indicating which peg is to be moved. The message TO: will then let the player know that the computer is waiting for a number to indicate where to move the peg. Notice that these messages are very short, but that they get the point across. You don't need to make the player read a book of instructions when a very short message can accomplish the same task.



When the user enters responses at the keyboard, you need to make it clear what the entry is to be for, and then check that the entry is valid. (This is not yet part of the legal game move check.) If a user is to enter a number, what happens when a letter or (ENTER) is pressed? The program must catch this sort of thing, and either allow the player to re-enter, or give the player some warning. This kind of a check is called "exception testing." It is one of the most overlooked aspects in programming. Often, when testing the program, the programmer knows what the correct type of entry should be, and never thinks about what would happen *if* . . .

There are two special values in *Peg Jump* which you can enter at the FROM: prompt. If you enter 88, the program will start the Auto Solve mode and actually show you how to solve the puzzle. The second special input is 99, which will reset the board to the beginning of a game. Use this option when you realize that there is no hope of completing the puzzle.

Check for Legal Moves

This section extends beyond simply verifying that input is of the right type. The player should not be able to do anything that would violate the rules of the game. If you rely on a player's honesty to keep from making illegal moves, a player could still make an unrecoverable mistake. Before writing this section of the code, you should devise the rules of the game. As an example, the rules for *Peg Jump* are listed here:

1. You must move from a hole containing a peg.
2. You must move to a hole which is empty.
3. All moves must jump one—and only one—hole.
4. The hole jumped must contain a peg.
5. The peg jumped must be removed after the jump.
6. Pegs can't jump diagonally.

Each of these cases should then be checked. If the check fails, the program returns to get a legal input. When designing your own games, you may wish to make the player lose a turn, or lose points for attempting to make an illegal move. Then the outcome of the game will depend to some extent on how well the player follows the rules.

Move the Peg

This section of the program takes a player's input, and changes the game situation accordingly. In this case, the computer must move a peg from one hole to another, and remove the peg jumped. It is difficult to pin this section down to DO's and DON'T's in general terms, because there are a large number of ways this situation could be handled.

If there is to be any action—or reaction—to a player's input, it should be handled in a way that is agreeable to the player. Taking *Peg Jump* as the example again: If all we did was erase the peg from one hole, draw it in another, and then erase the peg jumped, the real action of the program would be missed. The player has no concept of a peg jumping from one location to another. You should try to convey to the player what is taking place, not just display the results. Here, we were able to accomplish this with a little "low level" animation. By taking the peg to be moved, and actually showing it sliding across the board to its new location, the player gets a real feel for what is happening. In almost all games, the extra effort to produce realistic effects is worthwhile.

End-of-Game Check

All game programs should make a check at some point in the game to see if it is over. While the player may be aware that the game has come to a halt, the program must also be aware. If the game is over, or the end of one part in a multi-part game is reached, you must decide what to do. You should display the score, and possibly any other game statistics that may be interesting, such as the high score.

After the game is over, you should give the player a chance to play again. No program should just dump the player back to the system without asking permission—it's just not polite. If the player chooses to play the game again, there are a number of things you should keep in mind. First of all, you will probably need to re-initialize some of the variables, in particular the score. In doing this, try not to re-initialize more than is necessary. You shouldn't have to go all the way back to the first line of the program. In *Peg Jump* the location of each hole on the screen is stored in an array. This array doesn't need to be reloaded for each game, because it stays the same. This saves considerable time in the re-initialization phase of the program.

The second thing you should watch for is the re-entry of skill levels, or the like. There are no skill levels in *Peg Jump*, but many programs do contain them. If a player chooses to play again, you should decide whether he or she is allowed to choose a new level.

Auto-Solve Mode—A Bonus

One program feature usually found only in (but not limited to) puzzle-type games is an auto-solve mode. It enables the computer to solve and prove that an answer to the puzzle is possible. This type of feature is considered a bonus, and is not required to play the game. The level of intelligence you give the program depends solely on how much you want to dazzle the person using the game. In *Peg Jump*, we chose the simplest form of auto solve. No real intelligence is involved, so the computer doesn't need to actually find a solution—it has all of the moves required to solve the puzzle stored in DATA statements. Then it simply READs the next move, and uses the same routines that the player uses to move a peg.

"No program should just dump the player back to the system without asking permission—it's just not polite."

The Final Phase

After every program is written it should go through rigorous *play-testing*. This part of a program's development can't be overemphasized.

Because it is often difficult to be completely objective with your own program, you should have several other people unfamiliar with the program take a whack at it. It's even better if these people are unfamiliar with computers. Have them document anything unusual they might encounter. Many times you will write a program, and because you already know the program's or system's limitations, you will only design the program to the point where it does everything right. This is not enough. The program must also watch for anything the user might do wrong. This may be as simple as not letting the user type a letter when a number is expected, or making sure illegal control keys don't upset the game.

Try everything you can think of to make the program *bomb* (stop running prematurely). If the program survives the experience, you're getting closer. Until every line of code has been run with every possible combination of values, the program may still contain hidden bugs.

Have Fun

You can take a certain amount of pride in writing your own game programs. It really pays off when you see people playing one of your games and they say, "Wow, this game is great. Where did you get it?" You can just flash them a broad smile and say "I wrote it myself."

For *your* Key-in listing see HCM PROGRAM LISTINGS Contents on page 85.

CONTROL CAPSULE
Peg Jump

VALUE INPUT	FUNCTION
1 - 33	Move a peg from one hole to another.
88	Auto-Solve mode.
99	Reset the game.

The Apple *II* program uses shape tables to generate graphics shapes on the screen in BASIC. These shape tables were used to generate small numbers from 0 to 9 for the identification of holes on the board. The data for these shapes in contained in lines 1210 through 1410. Using these shapes in place of the standard characters offers several advantages: (1) They can be positioned anywhere on the screen. (2) They are not limited to the boundaries of the normal character. (3) They can be placed on the graphics screen. (4) They can be rotated or expanded using the ROT=, or the SCALE= functions.

The routine that uses these shapes resides in lines 1080 through 1160. It could be easily modified for use in any other application. Using these shapes would allow you to display up to 80 digits per line on a graphics screen.

Because of the Commodore's ability to print graphics characters within the quotes of a string, we decided to simply PRINT the board on the screen—rather than place it on the screen one character at a time. This is done in lines 480 through 700.

The title screen for this version was spruced up using the same method. Graphics characters were formatted within strings, and then PRINTed to produce enlarged text characters.

A FOR_NEXT loop is used to move a peg. This routine, in lines 980 through 1030, will move the peg across the board. Each time the peg advances one character position, the character below it is saved, so that after the peg leaves that spot, the original character can be returned. In doing this, the board's graphics are undisturbed by a peg moving across.

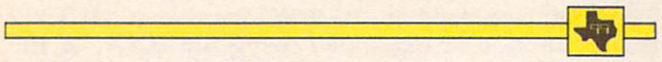
The IBM systems use quite a different method to create the playing board. If your PC system has the Color/Graphics Adapter and color monitor, or a compatible color system, you will be able to use this program, and the commands described below. If you have the PCjr, you will need only Cartridge BASIC.

The DRAW statement allows the programmer to draw lines and fill areas of the screen with color. This makes graphics creation extremely easy. For this reason we made the IBM board a little more elaborate by making it appear three-dimensional. It was a simple matter of using the DRAW statement to add sides to the board, giving it depth.

Animating the pegs posed a problem because the pegs interfered with the lines that connected each of the holes. Using the PUT statement to move a peg disrupted these lines if the peg started or stopped on a line. However, slanting the pegs got them away from the lines, and as an additional "bonus," remains consistent with the board's three-dimensional effect.

You may want to extract the small-numbers routine for your own use. These numbers are used to identify the holes. Each number is only four pixels wide. This would give you 80 digits across the screen on a 40-column screen, and 160 digits across on an

80-column screen. Each number is drawn with the DRAW statement. The commands to control the drawing are kept in a string array. There are ten elements, one for each number 0 through 9. This routine could be used in almost any program, with slight modifications. All you would need to do is pass the value to be drawn in the variable Z. The PSET command would then have to be modified to output the characters at the proper location on the screen. Currently, it uses an array of screen coordinates for the holes in the board.



The TI-99/4A version of *Peg Jump* makes use of character graphics to display the playing board. The program was written in BASIC.

To identify the holes, it was necessary to create small number labels to conserve space on the board, so that it will not look cluttered. These numbers are coded into ASCII characters 100 through 132. The FOR_NEXT loop in lines 540 to 570 reads the shapes for the numbers and assigns them to the characters. The loop in lines 950 to 970 then places these numbers on the screen in their appropriate locations.

To create the animation of moving the peg from one hole to another, two short routines were used. Lines 1660 through 1790 contain the two routines that will move the peg in any of the legal directions. Any characters that the peg passes over will be replaced, leaving the board's graphics unchanged.

Peg Jump (TI-99/4A) (TI BASIC) Explanation of the Program

Line Nos.	
100-180	Program header.
190-330	Initialize variables.
340-360	Data for game messages.
370-440	Display the title screen.
450-670	Initialize playing screen graphics.
680-1130	Display the playing screen.
1140-1230	Display current peg status.
1240-1470	Input the move from and the move to. Check preliminary limits.
1480-1660	Check for illegal moves.
1670-1820	Move the peg.
1830-1850	Update score.
1860-2090	Display "getting close" messages.
2100-2360	Display illegal move message.
2370-2380	Read next move when in Auto Solve mode.
2390-2400	Computer's victory message, after completion of the Auto Solve mode.
2410-2500	Music routine.
2510-2540	Routine to display text without scrolling the screen.
2550-2680	Input routine.
2690-2710	Data for peg locations.
2720-2820	Data for graphics characters.
2830-2840	Data for graphics positioning.
2850	Data for peg status display.
2860-2870	Data for Auto Solve mode.

Peg Jump (C-64) Explanation of the Program

Line Nos.	
100-180	Program header.
190-370	Display the title screen.
380-410	Store screen messages.
420-450	Initialize the game variables.
460-730	Display the playing screen.
740-920	Get move, and check to see that it's egal.
930-1090	Move pegs.
1100-1270	Routine to display messages.
1280-1380	Illegal move routine.
1390-1470	Clear message areas.
1480-1700	Music routine.
1710-1810	Input routine.
1820-1830	Cursor placement routine.
1840-2030	Data for the music routine.
2040-2090	Data for messages.
2100-2150	Data for peg locations.
2160-2190	Data for Auto Solve mode.

Peg Jump (IBM PC, PCjr) Explanation of the Program

Line Nos.	
100-200	Program header.
210-220	Display the title screen.
230-270	Initialize the program.
280-360	Draw the playing board.
370-440	Input moves and check for illegal moves.
450-480	Jump the peg vertically.
490-520	Jump the peg horizontally.
530	Display the peg status.
540-550	Subroutine to draw small numbers.
560-590	Key input routine.
600-650	Display the "getting close" messages.
660-680	Read next move for Auto Solve mode.
690-760	Display illegal jump messages.
770-790	Data for peg positions on the screen.
800-810	Data for the DRAW statement to create the small numbers.
820-830	Data for the "getting close" messages.
840-850	Data for the Auto Solve mode.

Peg Jump (Apple *II* Family) Explanation of the Program

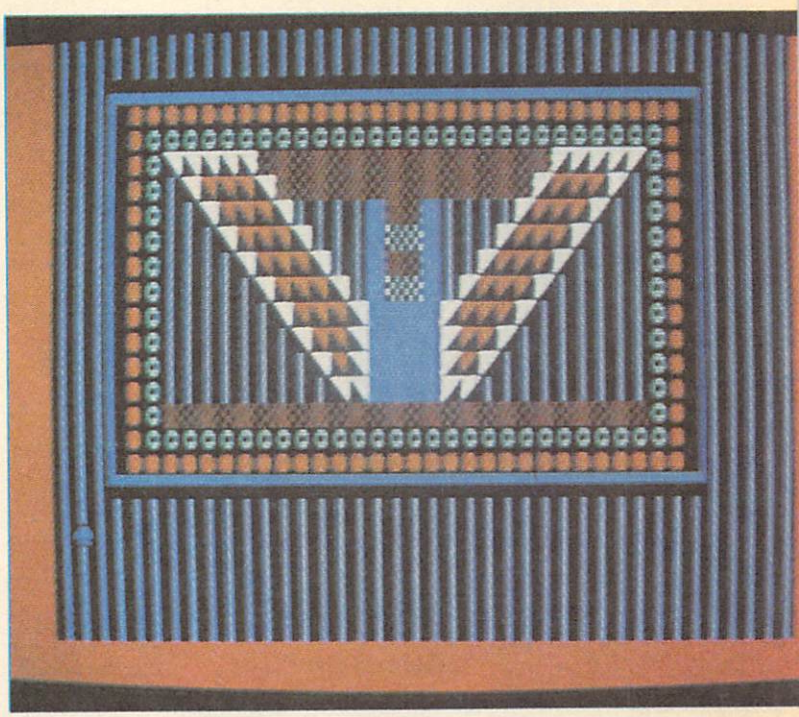
Line Nos.	
100-180	Program header.
190-380	Initialize the program variables and arrays.
390-400	Branch to POKE in the music routine and the shape tables.
410-420	Branch to set up the playing screen.
430-580	Input the FROM peg and the TO peg. Do preliminary limit checks.
590-810	Check for illegal moves.
820-930	Move the pegs.
940-1170	Auto Solve mode.
1180-1440	POKE shape tables into memory.
1450-1480	Call sound routine.
1490	POKE in sound routine.
1500-1550	Display "getting close" messages.
1560	Sound effects routine.

HCM

SKETCH-64

by James P. Chasse
and the HCM Staff

. . . designing and saving colorful graphic screens can be as easy as moving a joystick—and at times, twice as much fun.



What a masterpiece! All those hours of typing in PRINT commands, POKEing and PEEKing, and working with graph paper have finally paid off. A picture that once existed only in your imagination is now on the screen. Just look at what you can build out of the Commodore's graphics characters: colorful patterns, shapes, even shading. But after showing it off to family, friends, and other admirers, what can you do with it?

If you wish to save your artwork for posterity, or to visually enhance some other program, you face another complicated task—perhaps as difficult as creating the screen in the first place. For Commodore does not provide an easy means of saving and loading graphics screens.

Sketch-64 makes the joystick your link to Commodore graphics. You'll be able to select any of the Commodore graphics characters which belong to character set 1 (see Appendix E of your *Commodore User's Guide* for a complete list). This is the default character set, and because only one set is available at a time, *Sketch-64* always uses this character set. You can choose these characters from the keyboard, and then place them anywhere on the screen quickly and easily using the joystick.

"Commodore does not provide an easy means of saving and loading graphics screens."

Once you've created your screen, SAVE it to disk. Simply press the S key and your screen will be SAVEd to disk under the file name you've specified. You will even be able to print your masterpiece on paper! With *Sketch-64*, it's easy to create and SAVE a graphic work.

You can access these graphic files in your own programs by copying two short subroutines from *Sketch-64* into your program. The LOAD FILE subroutine (lines 1490 through 1700) prompts for a file name and then LOADs the file from disk into screen memory. This subroutine uses the second subroutine (lines 2160 through 2190) for disk error-handling. With these you can access as many files as you want and keep them all neatly and safely tucked away on disks.

Sketch-64 is ideal for constructing charts, menus, title screens, etc. Anything you can imagine drawing with Commodore 64 graphics characters can be created and SAVEd.

System Requirements

Commodore 64
Color TV or Monitor
Joystick attached to Port 2
Disk drive—if you wish to SAVE your drawings. (Note: it must be device #8)
Commodore (or compatible) printer—if you wish to PRINT your picture, or get a memory dump of the screen and color memory locations of your drawing.

Initial Options

With your joystick plugged into Port 2, LOAD *Sketch-64* and RUN the program. You will be presented with a series of questions and options. If the letter or number flashing under the cursor matches your response, simply press (RETURN), otherwise change it to what you want and then press (RETURN).

LOAD AN EXISTING FILE - This option is used to display or edit a screen you've created using *Sketch-64*. If you want to work on a brand new drawing, answer NO to this option by pressing (RETURN). If you want to work on a file you've already created with *Sketch-64*, type in Y and press (RETURN). In this case, the next prompt will be:

FILENAME - Type in the file name and then press (RETURN). Your file will automatically appear on the screen, you will be in drawing mode, and all of the other initial options listed below will be skipped.

If you do not choose to LOAD an existing file, then the following options will appear:

NEW BORDER COLOR - This option enables you to select the initial border color of your picture. You'll see the cursor blinking over a 1 (for white). You can change

this color by picking a different number between 0 and 15 (see Color Code Chart for the color values).

NEW SCREEN COLOR - This option works identically to the Border Color option, except it defaults to a 0 instead of a 1.

ASSIGN A NEW FILE NAME - If you intend to SAVE your screen, assign it a file name here. Unless you want to delete an existing file with the file name you choose, make sure that a file by that name does not exist. If it does, that file will be deleted and the new one SAVEd in its place. If you fail to initially assign a file name, you will still be able to assign one when you ask to SAVE with the S option (see Control Capsule below).

FILL SCREEN - This option allows you to fill your screen with any character from set 1 from the Screen Code chart (see *Commodore User's Guide*, pages 132-134). Filling the screen provides a means to create unique and interesting designs by carving, rather than filling. If you select Y you will be prompted for a character and color choice:

CHARACTER# - At this prompt, pick the number of the character you wish from the Screen Code chart.

CHARACTER COLOR - At this prompt, select the number of the color you wish from the Color Code chart.

Color Code Chart

0	Black	8	Orange
1	White	9	Brown
2	Red	10	Light Red
3	Cyan	11	Gray 1
4	Purple	12	Gray 2
5	Green	13	Light Green
6	Blue	14	Light Blue
7	Yellow	15	Gray 3

Keystroke Commands

The Control Capsule in this article designates the key commands available when drawing. Here are more extensive descriptions:

Fire Button: PICK A CHARACTER - The character selection here works the same as it does when using the keyboard. Press the fire button, and you will be prompted to select a character. Just type whatever character you wish to place on the screen. Then, by moving the joystick, you can place that character at many screen locations.

1: CHANGING CHARACTER COLORS - Press 1 until the desired color appears.

2: CHANGE BORDER COLOR - Press 2 until the desired border color appears.

3: CHANGE SCREEN COLOR - Press 3 until the desired screen color appears.

B: RESTART PROGRAM - If for any reason you want to go back and restart the program, simply press B. You will be given a chance to change your mind if you decide you want to SAVE your drawing before the program restarts.

E: ERASE SCREEN - (Remember "Erase.") Again, you will be asked if you are sure you want to erase your drawing before the computer erases it.

L: CALCULATE SCREEN LOCATIONS - *Printer required.* This routine LISTs character POKE values and character color of all the screen locations to a Commodore printer. When ready, press L (remember "Location"). You can then use this information to construct DATA statements in your programs to POKE the desired locations.

M: MOVING WITHOUT OVERWRITING - You may want to move across the screen without erasing or drawing. Simply press M (remember "Move"). Note that all other functions are disabled. To return to the graphic mode, press M again.

O: OUT OF REVERSE - Press O (remember "Out of reverse"), and the current character will be "un-reversed."

P: SAVE SCREEN TO PRINTER - *Printer required.* You can save your drawing to a printer. When ready, press P (remember "Printer"). Note that the aspect ratio of the screen is different from that of the printer, so your printed drawings will not be in the same proportion. Specifically, the characters on the screen are not square—they are taller than they are wide. On the printer, however, the characters are square. This means that your printout will tend to be "squashed" from top to bottom.

"You can access these graphic files in your own programs by copying two short subroutines into your program."

Q: QUIT PROGRAM - To end the program, press Q. The program contains a message that allows you to change your mind and continue with your drawing if you press Q by mistake.

R: REVERSE MODE - To get a reverse character, press R (remember "Reverse"), and the current character at the current screen location will reverse color.

S: SAVING THE SCREEN - When you're done drawing, press S to SAVE your screen (make sure your disk is formatted first). This SAVE will take about two minutes. It will, however, LOAD much faster.

Control Capsule
Sketch—64

KEY	FUNCTION
Fire Button -	Pick a character
1 -	Change character color
2 -	Change border color
3 -	Change screen color
B -	Restart program
E -	Erase screen
L -	Calculate screen location
M -	Move without erasing
O -	Out of reverse
P -	Print screen to printer
Q -	Quit
R -	Reverse mode on
S -	Save to disk

Sketch-64 Programming Techniques

Translating joystick movement into screen movement often presents many difficulties to a programmer. The joystick is actually a set of 5 switches: one for each of the four directions, and a fire button. When you select a direction you are actually closing a switch. When you select a direction like up and right, two switches are actually being selected.

The state of the 5 switches of the joystick plugged into port 2 is read by PEEKing location 56320. This location is a register on the 6526 Complex Interface Adapter chip. The lower 5 bits of this address reflect the state of the joystick switches. These lower 5 bits are set to 1 when a particular direction is *not* chosen or when the fire button is *not* pressed; the bit is cleared to zero when a direction *is* chosen.

Here's a short program that prints on the screen the current PEEK of location 56320, and then prints the values as they are converted in *Sketch-64*. Note that we've included an intermediate step that negates the current value, so that you can see how *Sketch-64* converts the relatively obscure value PEEKed from this location into more useful numbers:

```
100 REM *JOYSTICK DEMO*
110 PRINT"■SHIFT CLR■"
120 PRINT"■HOME■PEEK(56320)=■4SHIF
    T CRSR LEFT■";PEEK(56320)
130 PRINT"■CRSR DOWN■NOTPEEK(56320)=
    ■4SHIFT CRSR LEFT■";NOT(PEEK(56320))
140 PRINT"■CRSR DOWN■(NOTPEEK(56320))+12
    8=■4SHIFT CRSR LEFT■";(NOTPEEK(5
    6320))+128
150 FOR TD=1 TO 100:NEXT:GOTO 120
```

When you run this program with a joystick plugged into port 2, you'll find that the PEEK of memory location 56320 is converted from a value between 111 and 127 to a number between 1 and 16, depending on the state of the joystick. By experimenting with this program, you can find which converted value corresponds to each joystick positon. For example, when you press the fire button, the value becomes 16. If you move to the up position, the value becomes 1, down is 2, left is 4, and right is 8. Combinations of positions cause these values to add together. For example, up and left returns a value of 5.

In *Sketch-64*, the same program logic sets the D variable to the converted value. Lines 850-950 read the joystick plugged into port 2 and translate the PEEK value into proper POKE values to move the cursor.

```
850 D=(NOTPEEK(56320))+128
860 IFD=16THENGOSUB 960
870 X=SGN(DAND8)-SGN(DAND4)
880 Y=SGN(DAND2)-SGN(DAND1)
890 IF SC<1064 AND Y=-1 THEN SC=SC+1000
900 IF SC>1983 AND Y=1 THEN SC=SC-1000
910 IFINT((SC-1024)/40)=(SC-1024)/40 AN
    D X=-1 THEN SC=SC+40
920 IFINT(((SC+1)-1024)/40)=((SC+1)-102
    4)/40 AND X=1 THEN SC=SC-40
930 SC=SC+X+40*Y
940 CO=54272+SC
950 POKESC,CH:POKECO,CL:GOTO 580
```

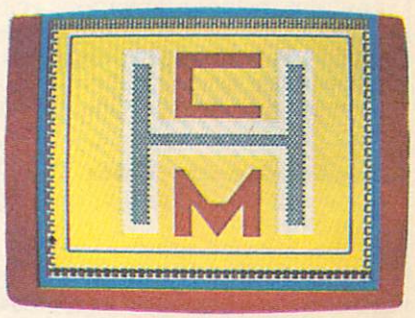
Line 850 PEEKs location 56320. D is then used in line 860 to cause a jump to the PICK CHAR subroutine in line 960 if the firebutton is pressed (D = 16). Lines 870 and 880 use the AND and SGN function to set X and Y to + 1, − 1, or 0, depending upon the values obtained. X and Y are used later in the program (lines 930-950) to update the POKE location for the active character according to the joystick's position.

Lines 890-920 are important because they cause the cursor to properly wrap from top to bottom and right to left. The SC variable is the cursor's current SCreen location. (It may be helpful to refer to the Screen Memory Map in Appendix G of your *Commodore User's Guide* for the following discussion.) If the cursor is in the top row (i.e. − SC < 1064), and the joystick is in the up position (Y = − 1), SC is increased by 1000 (the size of the entire screen memory). When line 930 updates SC, it will cause the cursor to appear in the same column in the bottom row of the screen. Similarly, if the cursor is in the bottom row (SC > 1983), and the joystick is in the down position (Y = 1), SC is decreased by 1000—so the cursor will appear in the top row. SC is also used to update the current color location in line 940.

The logic in lines 910-920 assures the side-to-side wrap. If the current cursor position (less the 1024 offset of screen memory) is evenly divisible by 40, and the joystick is in the left position (X = − 1), then line 910 increases SC by 40, so the cursor will appear on the right edge of the same row. Line 920 decreases SC by 40 when the cursor is in the right-most column and the joystick is pointing right (X = 1), thus causing the cursor to move to the far left column of the same row.

Enjoy Joystick Graphics

And there you have it. A very useful utility indeed. You could tailor this program to meet your own needs. You could add a program segment to redefine the character set, and create your own unique graphic screens. Or, if you want to include a great deal of text, you might develop a text mode. Be sure to keep us informed of your enhancements in a Letter to the Editor.



A relatively complex graphic screen can be created and edited in minutes using a joystick.

For *your* key-in listing see HCM PROGRAM LISTINGS Contents on page 85.

Sketch—64 (C-64) Explanation of the Program

Line Nos.	
100-170	Program header.
180-280	Dimension arrays and initialize variables.
290-570	Initial options.
580-950	Joystick and keyboard input routines.
960-1050	Pick new character routine.
1060-1100	Change screen and border color.
1110-1260	Save drawing to disk.
1270-1480	Screen calculations for disk.
1490-1700	Load file from disk.
1710-1860	Screen calculations for printer.
1870-2000	Cross screen without drawing.
2010-2070	No-file-name routine.
2080-2150	Display message and restore drawing.
2160-2190	Disk-error routine.

HCM

HCM ONE-LINERS

In the "Letters to the Editor" section of the previous issue, we put out the call for one-line programs written in any language on any of the computers that are covered in *Home Computer Magazine*. The response was so great that we decided to devote an entire page to one-liners. Although many interesting programs were submitted, we have selected what we felt were the four best (one for each brand of computer) of those that arrived prior to this issue's press date. If you have not submitted your masterpiece, it is not too late! As long as we keep getting great one-liners, we'll keep filling this page for you. Our prize winner this issue is G. A. Hamilton, who won $50 for sharing a unique one-line "arcade game" with our readers.

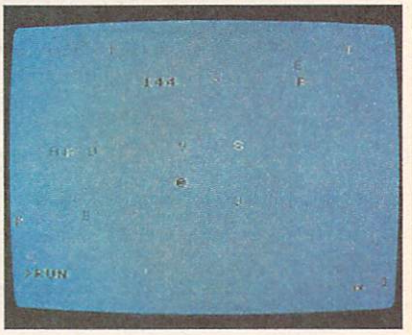
One-Line Arcade Game

[TI Extended BASIC on the TI-99/4A]
Dear Sir:
ALPHABET AT-TACK

You (the @ symbol) are being attacked by alpha characters. Points are scored against you for hits on your @ ship. Low score wins. The longer you take, the faster the alphas go. The program includes 24 various colored alphabet sprites that move in different speeds and directions. One sprite (@) is controlled by the No. 1 joystick. The score is continuously presented on the screen. Type until the computer beeps and press (ENTER). Type 1, then (FCTN) E, then (ENTER), then (FCTN) 8 and finish typing the line.

G. A. Hamilton
Nepean, Ontario, Canada

```
1 N=28::FOR X=4 TO N::C
ALL SX,RM COC:FX=X+4,C
AN,NALL M=M-CALMOT Y=Y+4,F,2
:C M:: CALL JOY(1,2*F,
9 :CALL NEXT Y::NEXT X
* E)::
```



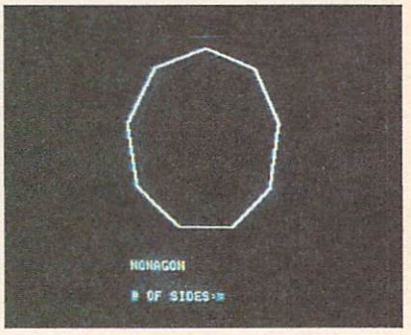
Apple Geometry Tutor

[Applesoft BASIC on the Apple II]
Dear Sir:

This one-line program draws and names the regular polygons of 3 to 9 sides. It must be entered with no spaces (except those between quotes) and using ? instead of PRINT. Note the use of RUN instead of the usual RESTORE: GOTO 1 sequence to save space.

Thomas Bavis
Macedon, NY 14502

```
1C=69:VTAB24:HCOLOR=3:?
:?:?"#OF  S IDES:HPLOT
HGR:P=  3.14/NSIN:HPLO
SIN(P),C+C*COS(P):HPLO
1TO:Q=P+2*I*P:HPLOT C+
+CD*A$:?:?A$:DRAW P,C
A?,AGOTAGON,HEXAGONAGON
N,OCTAGON,NONAGON
```



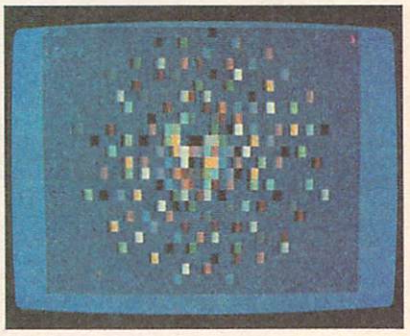
Graphics Spectacular

[Commodore BASIC on the C-64]
Dear Sir:

My submission creates a spiral of multi-colored squares—resulting in what appears to be an explosion of color. I used the abbreviated form of the BASIC keywords FOR [F (SHIFT) O] and POKE [P (SHIFT) O] in order to fit the entire program within one Commodore BASIC line.

Paul Kelley
Vancouver, BC, Canada

```
1 F=SHIFT O C=1T0 9 6STEP
.5:A=SIN(C)*40:POKE S
(C)*C/8)*40:POKE SHIFT O
524+A,160:POKE SHIFT O 55
96+A,C:NEXT
```



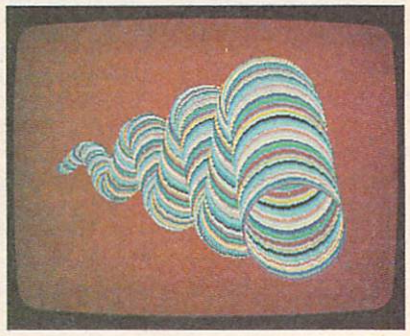
The Colorful Cosmic Worm

[Cartridge BASIC on the IBM PCjr]
Dear Sir:

Since I purchased my IBM PCjr it has convinced me that there is no other computer in its price range that can match its graphics capabilities. I created this one-line program as a demonstration to prove my point. I would like to challenge anyone with a non-PCjr system to duplicate the Cosmic Worm in one line.

William P. Scott
New York, NY 10028

```
1 CLS:KEY OFF:COLOR 5:CLEAR ,,,4
32 4:FOR Z=1 TO 2:C=C+.24:C=-C:NEXT
,4 .00,Z=FNT(Z*50)/10:PSET(+
200),T:PALETTE 0:C
0EXT,C:PALETTE 1,C
+C,TO
```



All One-Liner submissions are subject to the same publishing criteria as the Letters to the Editor (explained in the magazine's masthead on page 4). If you have written a great One-Liner in any language on any computer covered by *HCM*, send it addressed to: Letters to the Editor, 1500 Valley River Drive, Suite 250, Eugene OR 97401. You too may win a cash prize and be immortalized in print!

HCM Review Criteria

Each month, *Home Computer Magazine* (*HCM*) reviews products designed for the Apple *II* Family, Commodore 64 and VIC-20, IBM PC and PCjr, and Texas Instruments 99/4A computers. *HCM* reviews take a detailed look at the quality, utility, and value of commercially available packages for these machines. Because our publishing charter forbids accepting outside advertising, we strive to make the scope and content of our review pages shine with a unique blend of humanistic frankness and objectivity.

Not only will you find all relevant information for making a wise purchase decision, but in some special cases we also provide nuggets of compu-prestidigitation.* For example, we frequently include essential documentation not furnished by the manufacturer. Additionally, each issue of *HCM* tries to review at least one outstanding product —a "Diamond in the Rough"—which, because of company size, marketing clout, or for some other reason, has not received the attention it deserves.

At the beginning of each review, a review-at-a-glance box provides the user with an instant assessment of the product. Each item will be evaluated, where relevant, with the criteria below.

HCM Review

Name: Old Art
Program Type: Recycled Graphics
Machine: Apple *II* Family, C-64 & VIC-20, IBM PC & PCjr, TI-99/4A
Distributor: Hit 'n' RUN Software, Inc.
Price: $99.99 (or trade for '72 Pinto)

System Requirements:
Disk Drive, Joystick, Trash Can optional

	Poor	Fair	Good	Excellent
Performance:				▬
Engrossment:			▬	
Documentation:			▬	

*** Performance—**
How well the product performs as intended; how well it takes advantage of a specific machine's capabilities; how well it responds to the user's commands; how effectively the graphics, sound effects, music, or speech are integrated with the software.

*** Engrossment—**
Whether the game or activity has that intangible quality that holds players on the edge of their seats while the hours tick by unnoticed.

OR

*** Ease of Use—**
The degree to which a user can interact with the product without outside help; the ease and effectiveness of error-handling features; whether the actual reading level of the activity is appropriate for the suggested audience.

OR

*** Ease of Set-up—**
How well the product design facilitates easy installation.

*** Documentation—**
The quality of the printed matter that comes with the product; whether the instructions are clear and comprehensive; whether the machine configuration requirements are spelled out. Information such as how to load a program, use the keyboard, and restart an activity contributes to the documentation rating, as do tips on performance peculiarities.

Products may also be evaluated in the following areas:

*** Flexibility—**
Can the product be adapted to the specific needs of the users?

*** Cost/Benefit—**
Is the product worth the user's investment in time and money?

*** Necessity—**
Is the product a solution for which a problem already exists?

*** Originality—**
Is it unique in concept, or simply a "me too" product?

*** Longevity—**
The "Boredom Factor." Does the program sustain interest?

*** Rewards—**
Are the audio-visual rewards motivating and appropriate?

*** Concept Presentation—**
Are the concepts presented clearly, logically, and in depth?

*** Special Effects—**
How does quality of sound and visual effects rate? Do they enhance or detract from the product or learning process?

Attention Software Authors & Peripheral Inventors:

* WANT TO BE DISCOVERED? *

Home Computer Magazine Wants To Give You A Chance!

We are looking for home computer products that have not received the attention they deserve. Each month, we will be singling out one such package for special review. If you have a unique commercial product of exceptional quality—but your advertising and promotion budget has not allowed you to capture major media attention—we want to see it. We will consider reviewing any product that meets our high standards.

We are an Equal Opportunity Reviewer!

In order to qualify for possible review, your product must:

1. Currently be available for purchase to readers of this magazine.
2. Make a unique and important contribution to the home computer industry.
3. Be of outstanding merit, quality, and value.
4. Be consistent with the type of machines and products we normally cover.

If you feel that your product qualifies, mail it to:

Home Computer Magazine
Attn: Editorial Submissions
1500 Valley River Drive, Suite 250
Eugene, OR. 97401

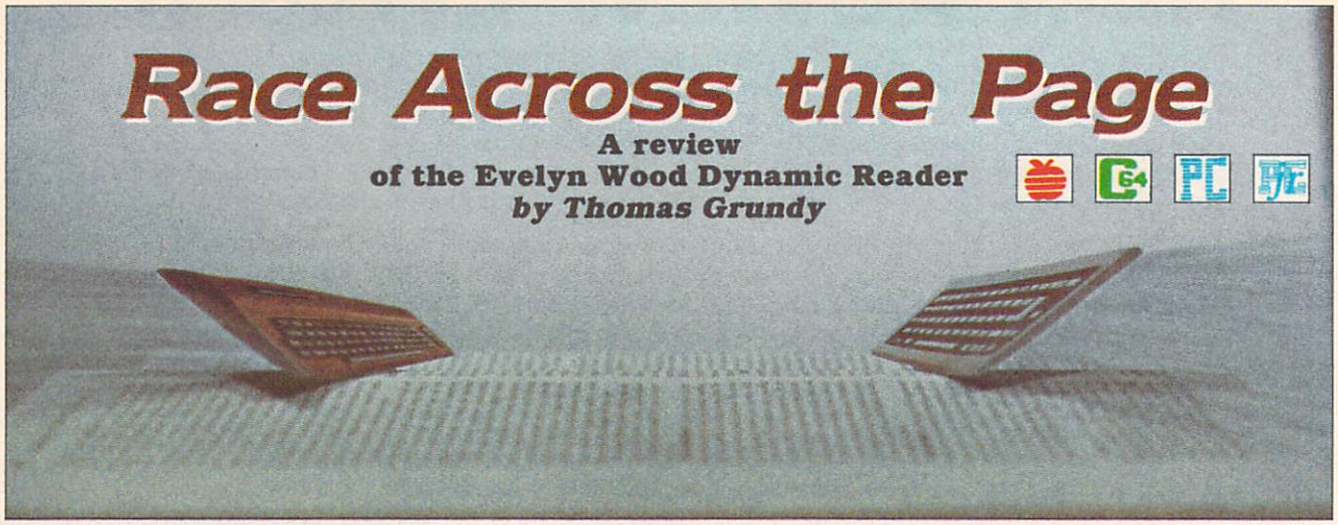
We reserve the right *not* to reply to each inquiry, so please do *not* contact us except to request return of your product. If you want your product to be returned, please include sufficient return postage.

***Compu-prestidigitation** (kóm´•pū•prĕs´•teh•dĭ•jeh•tā´•shŭn) —*n* 1. The magical quality of unexpected comprehension that results from presenting technical information about computers in a lively, entertaining, visually attractive and easy-to-understand format. 2. The magical tricks that make a computer sing, dance, and do all sorts of wonderfully useful things.

Race Across the Page

A review of the Evelyn Wood Dynamic Reader

by Thomas Grundy



If you really want to rev up your reading speed, can your computer get you in gear? Stay tuned . . .

Before I had even viewed the new *Evelyn Wood Dynamic Reader* program, my old bias began to stir, stretch its muscles, rear its ugly head, and roar. I mean, who would want to hurry through a poem by Yeats (or by any *good* poet) or a novel by Melville. Or a Shakespearean drama. Or an epic by Milton.

Satire spread its darksome wings and began to soar. I could title the review: "A Shakespearean Sonnet a Second," or, "How I Read *Hamlet* during my Fifteen Minute Juicy-Fruit-Gum Break." Or maybe: "Paradise Lost but Hours Gained," or, "Thumbing Your Way Through the Garden of Eden in A Week in Less Than Twenty Minutes a Day." A marvelous paradox presented itself. At the end of the second paragraph I would write: "By now, if you speed read, you've already finished this review."

I had hardly begun congratulating myself on my wit when I heard my all-too-familiar editorial voice rasping: "Be fair! Be fair! Remember both sides!"

It Would Be Nice

Okay! Okay! After all, you don't need to speed read everything. But it would be nice to get through newspaper and magazine articles—and even some books—a lot quicker. So, let's give this thing a try.

Before you begin, as with any new program, read the manual. My father says, "After all else fails, follow the directions." The documentation for this program is very clear and easy to follow. Setting up is a little more difficult for the IBM PC/PCjr than for the Commodore 64, because the IBM version requires copying DOS onto the program disk first. But after this one-time procedure is completed, the IBM program becomes relatively easy to use.

The program, although "not intended to be a substitute for the Evelyn Wood classroom class," still promises to "provide you with all the basic tools you need to become a Dynamic Reader,"—a "Dynamic Reader" being someone who reads quickly and with good comprehension. The "tools" that the program provides you with are divided into three groups: Skills, Drills, and Readings.

Before turning to the Drills, it is important to note that you *really* need a good computer monitor or one of those new-fangled televisions that has 10,000 lines per square inch. Otherwise, after you've finished a few exercises or readings on your Sears 19" color TV, you'll look like a ground hog on the first day of Spring, squinting and bumping into things. The eye strain is incredible. You may even need a different type of eye exercise after a month of trying to differentiate between the p's and g's, the u's and v's.

Exercising the Eye

The Skills section has four parts: characters, words, phrases, and eye exercises. The first three of these are designed to increase the speed at which you recognize letters, words, and phrases. For example, a word (or group of characters or phrase) is projected onto your screen. Below the "key word" is a list of words containing this word. You are to locate this "key word" as quickly as possible; the computer will time you. The only trouble that I had with these exercises was trying to find the appropriate number to press on the keyboard. Those of you who do not type by the hunt-and-peck method, however, will not have this problem. The eye exercises, which are meant to "train your eyes and hand to move at a rapid pace," are nice—sort of follow-the-bouncing-ball routines—but oddly enough the little ball (or dot of light) does not trace you through any readings. It just zips across a blank, black screen.

"Satire spread its darksome wings and began to soar. I could title the review: 'A Shakespearean Sonnet a Second,' or, 'How I Read Hamlet during my Fifteen Minute Juicy-Fruit-Gum Break."

The Drills, though tedious at times, promise to be effective if you keep working at them. The program gives you three types: the "push-down," the "push-up," and the "power" drill. For all of these drills, you pick a reading—any reading from a book, newspaper or magazine of your choice. To get some idea of how these drills work, let's look at one. How about a "push-down" drill? First you need a reading. Something light would be good. Sure, Kierkegaard's fine. Read as much as you can in a minute. The computer will time you. Now re-read the same material in 45 seconds. Now in 30. Now in 20. Next, in 30 seconds, read as much new material as quickly as you can. Now re-read this

material in one minute, this time for comprehension. Rub eyes. Stretch. Enter into the computer the number of words per line, lines per inch, and total number of inches read in the last minute. Answer the five self-evaluation questions the program gives you. Your word-per-minute (wpm) speed and comprehension percentage will appear on the screen. Thus, your computer serves as a timing clock and a calculator.

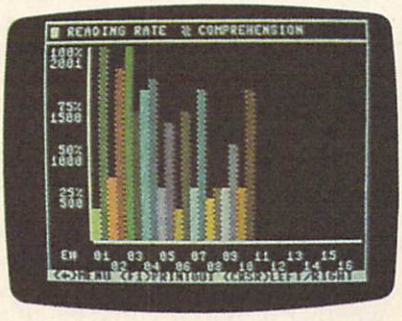
The Readings are, as the documentation points out, "the meat of things." You are provided with ten on-computer and ten off-computer readings, and two quizzes for each of these readings. The computer times you, giving your wpm speed; the quizzes test your

"Indeed, except for the window portion for on-computer readings, everything in the program could be done just as well without the aid of the computer."

comprehension. When you choose an on-computer reading, you are given the option of reading by either Full Page or Window. The Full Page option is exactly that: you may view one full page of text at a time. When you are ready to move on to the next page, you simply press a key. The Window option is perhaps the best part of the program. Just key in your pace (from 50 to 4000 wpm), and the computer flashes the reading on your screen, one sentence at a time, at whatever rate you've keyed in. Unlike the Full Page option, Window *forces* you to read at that predetermined rate.

Time To Taste?

Although some of the readings are rather "tasty" (like a short story by Mark Twain), I cannot help but observe that this "meat of things" has been sliced rather thin. Because there are two quizzes per reading, theoretically you are able to "use a reading more than once," but I found that this is not always the case. I performed an on-computer reading and took one quiz, reading at 300 wpm with 90% comprehension. One week later, as an experiment, I chose the same on-computer reading, but took the second quiz. I "flipped through the pages" *asfastaspossible* (but without re-reading any of the material), and voila-I still scored 90% on comprehension, but this time my "reading speed" was an amazing 4000 wpm. Needless to say, this was a test of my *memory*, not my reading *speed*.



A colorful graph depicts your progress in becoming a Dynamic Reader. (Photo from the C-64 version.)

In my estimation, providing two quizzes per reading does not effectively double the number of readings. And 20 readings are not that many. As the documentation points out, however, you may choose your own off-computer readings. But even though the computer may calculate your wpm speed (after you plug in the pertinent information), no "substantial" quiz on the material can be given (or taken). Self-rating questions are provided (as they are for the drills), but, this type of question is not always the most accurate.

Name:	The Evelyn Wood Dynamic Reader
Program Type:	Speed Reading Course
Machines:	IBM PC, PCjr, C-64 (Apple release soon)
Distributor:	Timeworks, Inc 405 Lake Cook Rd. Bldg. A Deerfield, IL 60015 312-948-9200
Price:	(IBM) $89.95; (C-64) $69.95; (Apple) $89.95
System Requirements:	Disk Drive

	Poor Fair Good Excellent
Performance:	
Ease of Use:	
Documentation:	
Cost/Benefit:	

After you've finished a few drills and readings, the Functions section will provide you with a nice graph and/or report to show you your progress both in reading speed and comprehension.

The *Evelyn Wood Dynamic Reader* provides you with the necessary tools to improve your reading skills. If you practice with the exercises, discipline yourself with the drills, and apply your newly-found skills to your reading, you will in all probability increase your wpm reading speed and your comprehension rate. And the Evelyn Wood people, in adapting their speed-reading course to the computer, offer the home-computer user some nice features, like the pull-down menu and the Window option for on-computer readings.

Asking For More

Indeed, the program has some value. It could, however, be a lot better. For example, it could offer more readings and make a few drills completely dependent on the computer. But more importantly, at least for the home-computer user, the program could be more fully adapted to the computer medium itself. The eye-exercise patterns could be, but have not been, integrated with the readings. If it is desirable (as the documentation maintains) to use your hand or finger as a pacer to trace and time your way across the written page, then it would make sense to integrate this function into the software's capabilities. On the Commodore 64 and IBM PCjr, for example, it would be a simple matter to add this visual stimulant and reinforcer to the program. The "bouncing ball" could thus lead you through the text. For computers with mice it would be nice to allow aspiring speed readers to chase themselves across a page at whatever rate their happy hands and eyes can carry them.

The *Evelyn Wood Dynamic Reader* still needs work—needs to make better use of the computer's potential. Indeed, except for the window portion for on-computer readings, everything in the program could be done just as well without the aid of the computer.

As it is, the software package will work—in all probability, you will become a faster reader if you stick to the program and practice, practice, practice. The package's weaknesses are not fundamental to the concept; rather, they are in the adaptation (or lack of it) of the speed-reading concept to the computer medium.

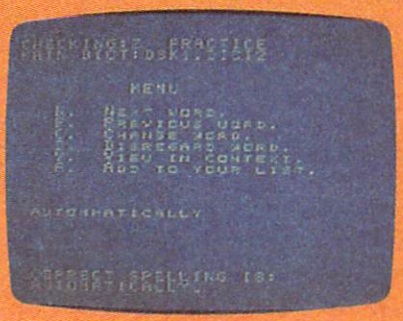
And, after considering everything, I have to admit that speed-reading does have its place. After all, I don't have to speed-read if I don't want to. But the next time someone thrusts the complete works of Rod McKuen in my lap and tells me that I just *have* to read them—well, at least, I'll have the option. **HCM**

99/4A Auto Spell Check

A review by Steve Nelson

HCM Staff

HCM Review



Name:	99/4A Auto Spell-Check
Program Type:	Utility
Machine:	TI-99/4A
Distributor:	Dragonslayer American Software Co. 2606 Ponderosa Dr. Omaha, NE 68123
Price:	$49.95
System Requirements:	32K, disk drive, TI-Writer cartridge or Editor Assembler cartridge.

	Poor	Fair	Good	Excellent
Performance:				
Ease of Use:				
Documentation:				

For help catching your misspelled words, check out this electronic spellchecker for the TI-99/4A. It's reliable. It's adaptable. But it's not exactly lightning fast.

Some people are lucky—they don't have a problem pelling words correctly. Other people, however, pell by trile and erur. And even the best spellers make misteaks once in a while.

In this computer-age automatic spell-checking programs have been developed, for most home computers, to help eliminate all spelling errors. But unfortunately, if you own a TI-99/4A, you have been out of luck—until now. Dragonslayer American Software Co. is offering a spelling checker for TI-99/4A owners called the *99/4A Auto Spell-Check*.

This program requires at least one disk drive, a 32K memory expansion, and a *TI-Writer* cartridge, or an *Editor Assembler* cartridge. (*99/4A Auto Spell-Check* will only check Display/Variable 80 formatted text files following *TI-Writer* or *Editor Assembler* character conventions.)

The main dictionary of 20,000 words, although somewhat small, is adequate when augmented by a user dictionary. *99/4A Auto Spell-Check* comes with an option called Seedgen, which lets you create user dictionaries of about 2,000 words each. This option is the program's best feature, because it allows you to tailor dictionaries to fit your specific word processing needs. For instance, if you use special words that are difficult to spell—such as medical or legal terminology—you can build a user dictionary of these terms, and use it to check your text.

How It Works

Auto Spell-Check scans your text, checking each word against the main dictionary, and stacking each word that it doesn't recognize into a "bad word stack." Once it completes its initial scan of your text, the program prompts you for a user dictionary name. It will check the words in the bad word stack against as many user dictionaries as you specify. After the program has completed checking your text against all dictionaries, you can review each remaining word in the bad word stack individually, and either correct the spelling, add it to your user dictionary, view the word in context, or disregard it. If you change the spelling of a word, it is flagged and—after you complete the review process—the program automatically updates your text file with the corrections. If you select a word for addition to your user dictionary, the dictionary is also updated. (You must have a user dictionary file already prepared in order to update it.) Because this program just stacks words that it doesn't recognize, it leaves it up to you to look up and key in the correct spellings before it updates your text file.

Problems

Unfortunately, *Auto Spell-Check* has some real problems. The execution of the program is slow—slow enough that if your text length is small, like a single-page letter, you could probably proof it yourself and make corrections faster than the *99/4A Auto Spell-Check* could do it (assuming you can recognize a misspelled word).

If you are working with only one disk drive, the constant swapping of disks can be quite cumbersome. You can, however, cut the number of disk changes required if you have a multi-drive system.

". . . an option called Seedgen lets you create user dictionaries of about 2,000 words each."

If a corrected word is not equal in length to the word it is replacing when the program updates your text, the program deletes the misspelled word, breaks the sentence, and inserts the new word on the line below. This means that you may have to reformat your text after it has been updated—somewhat of an inconvenience, to say the least . . .

After using *99/4A Auto Spell-Check* for a couple of weeks, I found myself resenting its relatively slow operating speed. But because the TI-99/4A is limited to 48K (even with the 32K memory expansion), its slow speed can be attributed to the program having to make frequent accesses of the disk. Its slowness, however, is something that should be considered when deciding whether or not to purchase this product.

If your typing skills are a bit on the sloppy side, or if you handle so much text that you can never catch every error, then the *99/4A Auto Spell-Check* may be worth looking at—provided you don't mind the product's limitations. **HCM**

simon sez:

Lessons on Using Simon's BASIC

by William K. Balthrop
HCM Staff

In a continuing effort to enlighten and brighten your computing skills, we bring you SIMON SEZ. This column is dedicated to people interested in Simon's Basic for the C-64. Simon's Basic adds 114 new programming commands, making the art of creating software much less painful.

Use the INSERT command to insert one string into another.

```
PRINT INSERT("RED","THE DOG IS BIG",4)
THE RED DOG IS BIG
```

The INSERT command allows you to insert one string into another. In this example, the letters RED will be inserted after the fourth character position—i.e., the space following the word THE. Try entering the example above, changing the word RED in the first parameter to REAL, and the number in the third parameter to 11.

Use the INST command to overlay one string onto another string, *overwriting* the original text.

```
PRINT INST("RED","THE DOG WAS OLD",12)
THE DOG WAS RED
```

The INST command is used to replace a portion of a string with a substring. The value used for the third parameter is a number that indicates the character position immediately in front of the characters to be replaced. Try entering the example above, changing the number 12 to 0.

Use the PLACE command to find the position of a substring within a larger string.

```
PRINT PLACE("RED","THE DOG WAS RED")
13
```

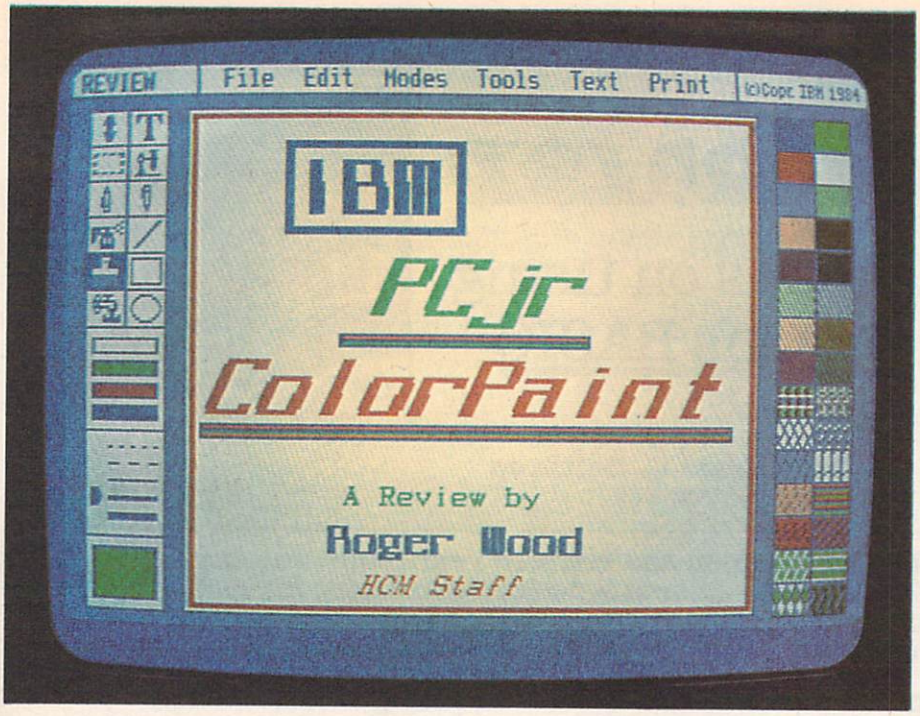
The PLACE command can be used to locate a string of characters within a larger string. The value returned represents the position of the *first* character in the substring within the larger string. If the substring can't be found in the larger string, or the second parameter is smaller than the first parameter, then the value returned will be 0 (zero). Try entering the example above, changing the word RED in the first parameter to DOG.

To duplicate or repeat a string over and over again, use the DUP command.

```
PRINT DUP("HO ",3)
HO HO HO
```

The DUP command is very useful in saving program space when a string must be repeated many times. Its only limitation is that the result must not excede 255 characters, or you will receive an error message on the screen. Try entering the example shown above without the trailing space after the HO.

HCM



Put away your water colors and latch onto a mouse! IBM's new painting system for the PCjr transforms your artistic visions into video reality.

The IBM PCjr has graphic potential beyond any machine in its price range. Its built-in high-resolution graphics capability (640 columns by 200 rows) in 4 colors is more than double the built-in horizontal resolution in Apple *II* Family, TI-99/4A, or C-64 computers. Now *PCjr ColorPaint* from IBM gives users an easy-to-use package to access Junior's built-in graphic power.

PCjr ColorPaint requires a mouse controller that has at least two buttons. People who are used to single-button mice may find the two-button *PCjr ColorPaint* commands to be a bit awkward. (For details on mouse compatibility, see "Special Setup Considerations" on the next page.) It's too bad that the product doesn't work with a joystick, as the cost of adding a mouse (from $195 to $495) will tend to discourage consumers.

In addition, television and composite video monitors are not adequate for the high-resolution display, so the user must have a relatively high-priced display to take full advantage of *PCjr ColorPaint*. Unless you have another use for a mouse and an RGB monitor, these requirements make this package inordinately expensive.

Not Just *MacPaint* in Color

A glance at the main screen immediately suggests Apple's *MacPaint* to anyone familiar with that product. Although *PCjr ColorPaint* isn't quite as responsive (in terms of tracking mouse movements) as its Macintosh counterpart, the PCjr's color edge makes this product an exciting entry into the market.

PCjr ColorPaint's tools are quite similar to those in *MacPaint*. The paintbrush, straight edge, circle, square, text-mode, fat-bits, etc. are all included in this fine package. But this isn't just a copy of the Apple product—it has its own unique features, and its ease-of-use rivals the Macintosh.

Take a look at the Help screen, accessible from the Tools menu (Photo 1). The left side of the screen shows four sections: the icons, the color indicators, the line widths, and the block showing the current pattern in use, which is selected from the menu of patterns on the right side of the screen.

Across the top of the screen are 6 pull-down menus. The File menu is your link with DOS, allowing you to Get and Save your pictures, Delete the files from disk, or Quit *PCjr ColorPaint* to return to DOS.

The Edit menu offers a full complement of editing functions—from Undoing what you drew last and Clearing the screen, to "flipping" a selected area horizontally or vertically. The Merge option, a unique feature, lets you merge a previously saved picture *behind* a selected section on the present screen.

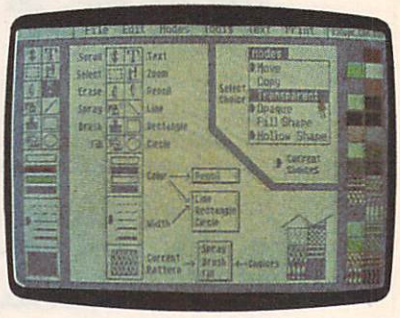


Photo 1
The Help screen available from the Tools menu.

The Mode menu works in conjunction with the Edit menu to let you either Move or Copy a selected portion from one area of a picture to another. Another Mode option lets you choose whether the white parts of a section being moved will be opaque or transparent in the new location.

The Tools menu lets you select brush shapes or edit the patterns available from the right-hand menu.

The Text menu gives you 4 different typefaces (or fonts) in 3 different sizes and 4 styles (i.e., italic, bold, etc.). This is one area where *MacPaint* is markedly superior—with many more fonts, sizes and styles, plus the added option of justifying your text.

Hi-Res Is *All* You Get

At first you might think that hi-res is all you'd want, but a close look at the PCjr's graphic modes raises doubts. Among the PCjr's graphic modes are high resolution (hi-res) mode, capable of 4 colors at a time, and medium-resolution mode, which offers 16.

Special Setup Considerations

We tried Mouse System's *PC Mouse* and the *Microsoft Mouse* (the two supported by the IBM package) and found them both to be satisfactory with no appreciable difference in response. Mouse System's *PC Mouse* attaches to the serial port (you'll need IBM's optional PCjr serial-port adapter to use this mouse on your Junior). The *Microsoft Mouse* can plug into either its own port on the *Microsoft PCjr Booster* memory unit or, with a series of adapters, into the serial port.

ColorPaint Without A Disk?

With the *ColorPaint* cartridge plugged into one of the slots in front, and your mouse connected and configured (each mouse package comes with special software so your system can use the mouse), you simply type G to get started. The *ColorPaint* documentation states that without a disk drive you can just turn on the PCjr with the cartridge in place and use the product. We found this to be true—but only with the mouse plugged into the serial port. Also, the disk controller card has to be removed from the system, because leaving the disk drive in a "not ready" condition (i.e., with the handle up on the drive) brings up Cassette BASIC instead of *PCjr ColorPaint*.

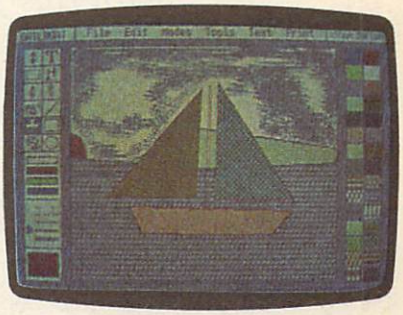
When the *Microsoft Mouse* is used with the *PCjr Booster* memory, MS-DOS is required for setup, so this mouse configuration doesn't work even when the disk-controller card is removed. *PCjr ColorPaint* requires 128K memory, which is usually accompanied by a disk drive on the PCjr. Disk storage is also the only way to save your pictures for future editing. We're therefore mystified as to why anyone would want to use the product without a disk drive, as suggested in the documentation.

Name:	PCjr Color Paint
Program Type:	Graphics Utility
Machine:	IBM PCjr
Authors:	Marek and Rafal Krepeo Inc.
Distributor:	IBM Boca Raton, Fl 33432
Price:	$99

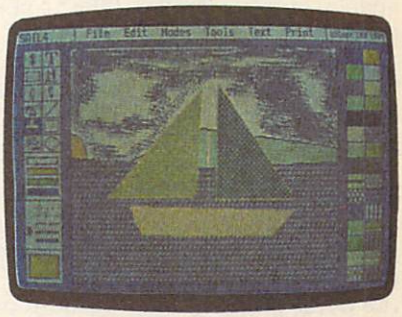
System Requirements: IBM PCjr w/128K bytes of memory, IBM PCjr-compatible serial interface mouse, IBM Color Display or IBM PCjr Color Display or equivalent (i.e.-must be Red/Green/Blue/Intensity (RGBI(Direct-Drive display.)

	Poor	Fair	Good	Excellent
Performance:				
Ease of Use:				
Documentation:				
Cost/Benefit:				

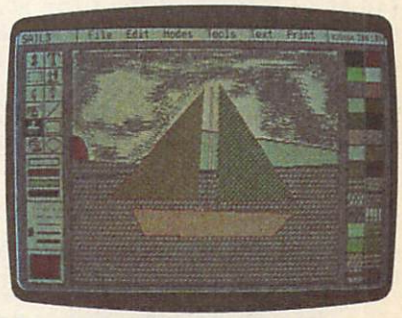
"At first you might think that hi-res is all you'd want, but a close look at the PCjr's graphic modes raises doubts."



Using only hi-res severely limits color choice: Photo 2 is a sailboat drawn with red, blue, and green.



In Photo 3, making the sun yellow changes the sailboat's colors.



In Photo 4, trying to change the sky to grey turns the water grey, too!

For maximum resolution, *PCjr ColorPaint* uses only the hi-res mode—but this increased resolution severely limits color choices. If the programmers had given the user the option of medium resolution mode, then all 16 colors would be available at one time, allowing for more diverse color choices in any painting. This lack of choice is a major drawback because color is the package's main strength.

Although it is impossible to display all available colors in one drawing, *PCjr ColorPaint* does offer the option of several *palettes* in hi-res. Thus, you can select any of the 16 colors, but only 3 (plus white, which is always the fourth) at a time.

Photos 2, 3, and 4 illustrate what happens when you try to change one color to make the drawing more diverse. In Photo 3, changing the sun from red to yellow has changed the boat's color. In Photo 4, changing the sky to a more somber color has affected the water's color as well.

The BASIC Picture

We had hoped that *PCjr ColorPaint* might allow us to use our pictures in BASIC programs or other applications, but this does not appear to be the case. By inspecting the files from DOS, we found that they are 28,672-byte DOS files with an .ART extension, but no technical information is provided about these files.

The package, however, is not totally limited to displaying on the video screen or saving to disk. If you have an IBM-compatible printer that will do graphics, such as the IBM Compact, IBM Graphics, or IBM Color printers, you can put your picture on paper.

Documentation

Two small manuals come with the product: one is a colorful, 38-page, tutorial-style book; the other is a reference manual that describes the various drawing tools in detail. These booklets are easy to understand, and are complete in terms of how to create pictures—but, as mentioned above, there is a distinct lack of technical information about the program.

Aside from the inability to use your drawings in other applications, and the above mentioned limitations on color selection, *PCjr ColorPaint* is a fine package. It lets anyone, novice or expert, tap the excellent graphic capabilities of the PCjr in a fun and easy-to-use way. The PCjr really gets to flex its graphic muscles when running *PCjr ColorPaint*. HCM

Graphics Magic At Your Fingertips

A Review of Super Sketch
by Steve Nelson
HCM Staff

It's a bit awkward—but you can draw detailed pictures with this graphics tablet for the TI-99/4A and C-64.

In the past, extensive programming was necessary to produce graphic images on a computer screen. Today, hardware/software companies are creating many different products designed to make computer graphics as easy and as familiar as drawing on paper.

Generally, computer graphics can be created 3 ways: (1) by brute-force programming (generating preplanned graphics screens with a language like BASIC), (2) using graphics software tools (programs that make it easy to build preplanned screens), and (3) using special input devices (light-pens, mice, joysticks, tablets, or sketch pads).

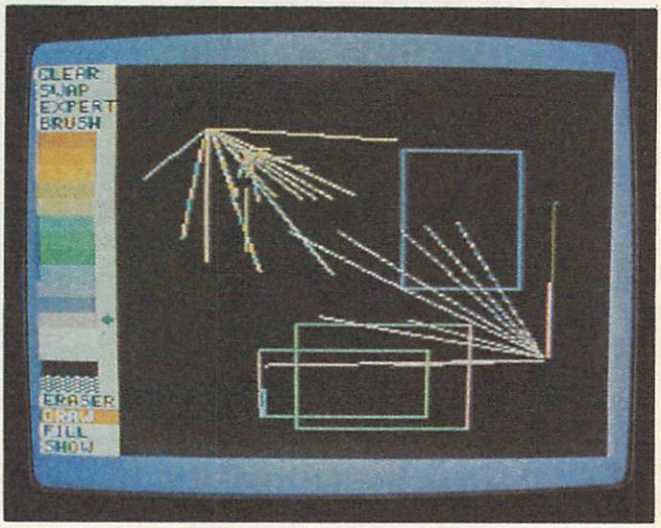
Super Sketch from Personal Peripherals is of the third variety—a sketch pad for the home computer, complete with graphics software included in the package.

Drawing

Sketch pads combine software and hardware to create a *frame buffer* that corresponds directly with the grid of the monitor screen. When you draw on the tablet or pad, its coordinates are registered in the buffer and transfered to the screen. The computer receives an X, Y coordinate and plots a "pixel" (picture element) on the screen in the color that you have selected. The software that controls this also lets you change colors, manipulate the brush size, and choose from a wide variety of options when creating your graphic design.

"My first attempt to trace even the most simple of designs—the bluebird picture supplied with the package—came out looking like a reject from a finish-the-drawing contest."

Unfortunately, the image that actually appears on the screen falls prey to the same problem that almost all computer graphics programs succumb to—"blocking." The line you draw is thicker than intended, or jagged around the edges. This happens because of the nature of digital computers, which "draw" on the monitor screen by creating a graph made up of horizontal rows and vertical columns. Each tiny square is called a pixel. Each pixel on the grid is represented by an X, Y coordinate. The computer "turns on" each pixel as directed by the software, and assigns it the color that you have selected. Occasional blocking results, even with the highest quality monitors, due to the digital signal nature of the signal.



Some of the "expert functions" on the TI-99/4A.

User-Friendliness is More Than Sketchy

Super Sketch is a hard plastic tablet with a stylus-tipped mechanical arm that you move to draw or trace. Its plug-in cartridges and fingertip controls make it very user-friendly. The two versions of *Super Sketch* software (for the TI-99/4A and Commodore 64) are quite different—the C-64 version is much more flexible, offering several options that are unavailable on the TI version.

For instance, the C-64 version has added features like the ability to copy portions of your picture to other areas, a window option, and even a mirror function which allows you to draw on one part of the screen while the program mirrors your drawing on another part. The most notable difference between the two versions is their method of storing pictures. The C-64 allows you to save to either disk or tape; the TI version lets you save only to tape.

The documentation for both versions is very good, with step-by-step instructions that take you through each individual function.

Performance

Setting up is a snap, and with a brief look at the instructions, you can begin drawing right away. The first thing you will notice about *Super Sketch* is the stickiness of its drawing arm, making it very

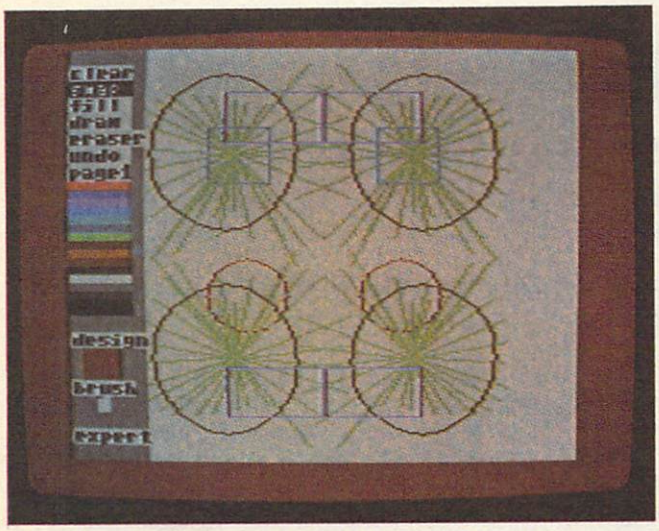
difficult to draw a smooth line freehand. This problem becomes especially evident when tracing. My first attempt to trace even the most simple of designs—the bluebird picture supplied with the package—came out looking like a reject from a finish-the-drawing contest. I find this problem to be a real drawback (no pun intended), as tracing should be one of the things sketch pads do best.

The response of the cursor—which indicates the position of the stylus in relation to the screen picture—is good when you draw with a slow, deliberate motion. If, however, you like to just scribble

HCM Review

Name:	Super Sketch
Program Type:	Graphics Tablet
Machines:	TI-99/4A, C-64 (Apple II and IBM PCjr versions were not available in time for this review)
Distributor:	Personal Peripherals 930 N. Beltline Suite 120 Irving, TX 75061
Price:	$59.95

	Poor	Fair	Good	Excellent
Performance:				
Ease of Use:				
Documentation:				



Some of the "expert functions" on the C-64 version.

away freehand, drawing as the spirit moves you, the cursor lags noticeably—especially when using a thick brush style. This is a real problem, as it can lag far enough behind to actually lose track of your drawing. There is also another problem with the cursor—it is too large. When you are drawing with a fine line, or using the Eraser or Fill functions to touch up tiny portions of your picture, the cursor can actually obscure your view.

All of the special features of *Super Sketch* can be accessed from the main menu. These include choosing colors (the C-64 version has 16 different colors, and the TI-99/4A has 15), changing brush sizes, filling your drawing with color, adding texture, and erasing. Selection of an "expert" menu helps you draw straight lines, boxes, circles, and rays—and, on the C-64, accesses the added features already mentioned.

"Setting up is a snap, and with a brief look at the instructions, you can begin drawing right away."

Color Me Dangerous

Some of these special functions work better than others. For instance, the Fill function can be treacherous. This feature allows you to select an area of your drawing and fill it with a certain color. You must, however, be certain not to have any gaps in the borderline around the area to be filled; otherwise, the fill color will spill over into the surrounding color and ruin your picture.

If you have a lot of intersecting lines in your drawing, the Fill option can be tricky here as well. If, in the process of adding color to your picture, the cursor touches a line, it can cause the line to change color. This in turn can cause every other line connected to it to also change color. It can even cause the entire picture to change color! (The C-64 version has an Undo option which will bring the screen back to the condition it was in before you made a mistake. This function is sorely missed on the TI-99/4A version.)

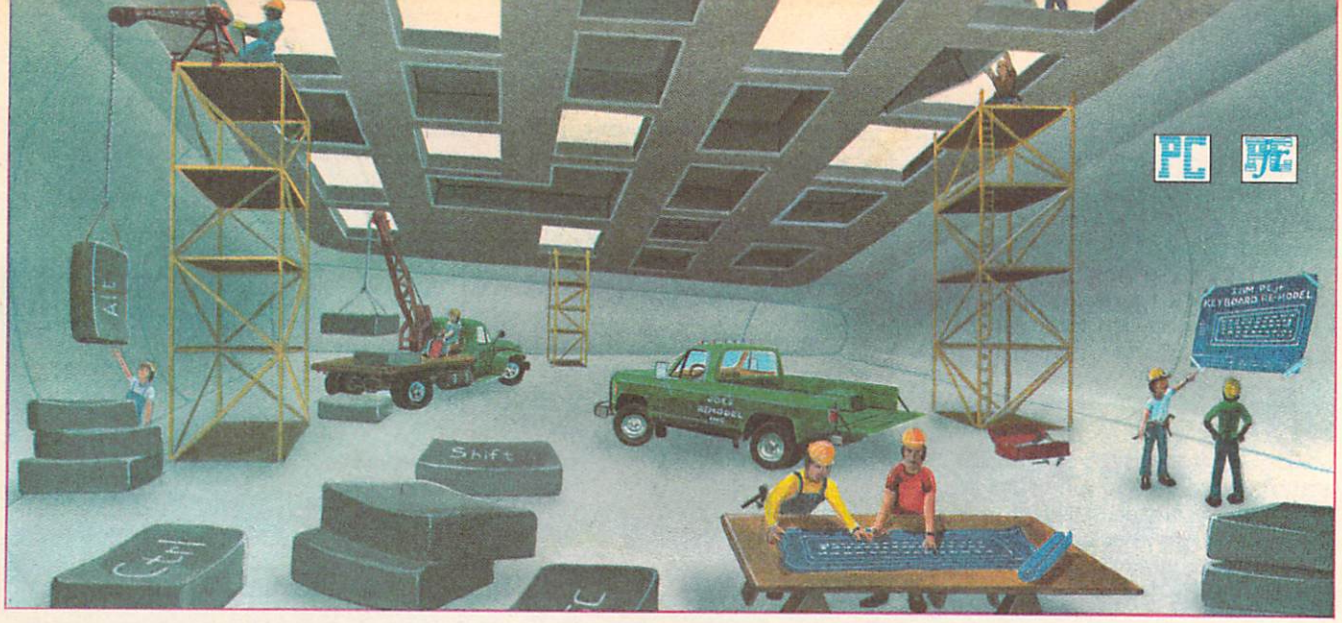
Drawing with *Super Sketch* may seem awkward at first, but with practice and patience you really can create spectacular graphics. The variety of bright colors and the special features of *Super Sketch* give you a powerful graphics package for the money. In addition, *Super Sketch* is very simple to use—especially for those who have little or no experience with computers.

Although I'm certain that there are people out there who trace with this sketch pad and are satisfied with its performance in that capacity, I certainly cannot recommend it for that purpose alone. As an artist's tool, the TI-99/4A version lacks flexibility, and both versions are much too clumsy to give reliable results.

However, as most graphics programs developed for the TI-99/4A have been programming aids rather than drawing tools, *Super Sketch* does offer an inexpensive and viable alternative. And although it's awkward to use, it does give you the ability to draw on your monitor without using the keyboard.

C-64 owners have a variety of graphics packages to choose from, including touch pads and light pens. The extra functions available on the C-64 version make *Super Sketch* very competitive with other graphics systems designed for this machine. [Peripheral Products plans to release a "souped-up" version (*Super Sketch II*) for the IBM PC and PCjr, and the Apple *II* family in early November—Ed.]

This graphics tool will not satisfy every budding computer artist. But some may accept the medium for what it is, anticipating its effects as part of their design. If you have been looking for something like *Super Sketch*, by all means try it before you buy it. **HCM**



New Keys for Junior

A Review of Three Keyboards for the PCjr

by Judy Campbell
HCM Staff

Two third-party keyboards for PCjr now compete with a free replacement from IBM. Is it a fair match?

Since it was first released, the original PCjr keyboard has come under heavy fire—the subject of derision from both new owners and the industry at large. Its chicklet keys made it seem more like a toy than a real keyboard, and this has reflected poorly on Junior's worthiness to its Big Blue pedigree.

New Boards Rush In

IBM recently moved to correct this situation by replacing the old board with a new improved model, but not before other manufacturers had rushed to fill the gap with their own PCjr-compatible units. Keytronic Corp. was first, with their model appearing before the new IBM release. Cherry Electrical Products was next—premiering their Model KFN3-9451 hot on the heels of IBM's new Junior keyboard (although it had been announced *before* the new IBM model appeared).

IBM is offering their new keyboard free to present PCjr owners, and as the standard for all future Juniors. Now the question is: Are these two new third-party models superior enough to IBM's release to warrant the added cost of purchasing either one?

Although the new IBM PCjr keyboard has banished the chicklet keys, the case is virtually identical to the old one. The new keycaps are shaped the same as those on the PC keyboard, and their touch feels more natural and familiar. The appearance of the new keyboard is more like what we're accustomed to seeing from IBM.

The new IBM PCjr keyboards are now available in computer stores. In order for PCjr owners to obtain their new IBM replacement, they can either take the old keyboard into the computer store and physically trade it, or they can take in proof of purchase and keep the old keyboard as well. There has been some concern that with the issue of the new keyboards, the software that requires special overlays for the old PCjr keyboard will fall by the wayside and be unuseable. But with the option of keeping the old keyboard, users who have purchased programs requiring overlays will not have to leave valuable software sitting on the shelf.

The Keytronic keyboard is much larger than both the old and the new PCjr board, which could be either a plus or a minus. It gives superior performance and greater versatility, but increases the desk space necessary to use it. The Cherry keyboard is a bit smaller.

Comparative Sizes:

Old PCjr keyboard:	13-1/4" x 6-1/2"
New PCjr keyboard:	13-1/4" x 6-1/2"
PC keyboard:	18" x 7-5/8"
Cherry keyboard:	17-3/4" x 7-5/8"
Keytronic keyboard:	20-1/4" x 8-1/2"

Both the Keytronic and Cherry keyboards are easy to set up. The Keytronic plugs into the PCjr. The Cherry, like IBM's own keyboards, can be used in two ways—either by plugging it into the PCjr, or by not plugging it in and using the two infrared links on the back of the board, freeing it from the computer and desktop.

Viva La Difference!

More obvious differences are evident between the Keytronic and the Cherry keyboards. We tested both boards with touch-typing and programming in mind. Even with large hands and long fingers, I found it difficult to stretch over to the (RETURN) key on the Cherry. It's much easier to reach the (RETURN) on the Keytronic (or even the chicklet PCjr board)—perhaps it's because the (RETURN) keys on the other two boards are broad on top, whereas on the Cherry the key is very small—dropping down to a broad base (which can cause the finger to slip off). Keytronic's positioning of the (CAPS LOCK), (SHIFT), and (ALT) keys are much better than on the Cherry or the PCjr boards. If you are a touch-typist, it's important for the keys

Name:	IBM PCjr Keyboard
Product Type:	Keyboard
Machine:	IBM PCjr
Distributor:	IBM Corporation P.O. Box 2328 Menlo Park, CA 94025
Price:	Replacement free

	Poor	Fair	Good	Excellent
Performance:				
Ease of set-up:				
Documentation:				

Name:	Cherry KFN3-8451
Product Type:	Keyboard
Machine:	IBM PC or PCjr
Distributor:	Cherry Electrical Products Corp. 3600 Sunset Avenue Waukegan, IL 60087
Price:	$235.00

	Poor	Fair	Good	Excellent
Performance:				
Ease of set-up:				
Documentation:				

Name:	Keytronic KB5151jr
Product Type:	Keyboard
Machine:	IBM PCjr
Distributor:	Keytronic Corporation P.O. Box 14687 Spokane, WA 99214
Price:	$225.00

	Poor	Fair	Good	Excellent
Performance:				
Ease of set-up:				
Documentation:				

to be in their close-to-normal (typewriter) positions. Also, the more responsive touch (2 oz. touch pressure) of the Keytronic is much more desirable.

Other very attractive features of the Keytronic are its LED on/off indicators for the (CAPS LOCK), (NUM LOCK), and (CURSR PAD) keys. It is aggravating to press the left cursor key—only to find you have left a string of fours, instead of back-spacing. The LEDs help solve this problem. (Even though the documentation for the Cherry model listed indicator lights for its (ALPHA LOCK) and (NUM LOCK) keys, there were none on the model we reviewed.)

Both the Keytronic and Cherry keyboards were supposedly designed with the option of using them on the lap. In testing, however, we found both awkward to use in this manner.

Bugs In Paradise

Although most features on the Keytronic keyboard are excellent, the board is not bug-free. Its documentation explains that the status of the (CAPS LOCK), (NUM LOCK), and (CURSR PAD) keys are maintained in the system RAM. However, some applications are known to "over-write" these locations, which causes the LED feature to be "out-of-sync" with the computer. A spokesperson for Keytronic said that developers of software applications are not supposed to change these bits, but some applications do change them, causing the "out-of-sync" problem. Specifically, the (CAPS LOCK) LED light can be on, and the computer will still be printing small letters on the screen. Keytronics recognizes this problem, and in the documentation they explain how it can be corrected: Holding down the (RESET) key at the upper right of the keyboard and simultaneously pressing the key with the errant LED solves the problem—putting the keyboard and the computer back in sync. We found that attempting to select a mode when the PCjr was accessing a disk also causes the problem, but the (RESET) key fixes it here as well.

A second problem occurs on the Keytronic when you activate the (CURSR PAD) key, which also automatically activates the numeric pad. This bug is due to a defective ROM chip in the keyboard. When in this mode, if you type a number using the upper row of number keys instead of the numeric key pad, the cursor pad ceases to function properly and begins printing numbers instead of moving the cursor. By experimenting, we found that pressing the backspace key or any of the numeric pad keys returned the cursor pad to its proper functions.

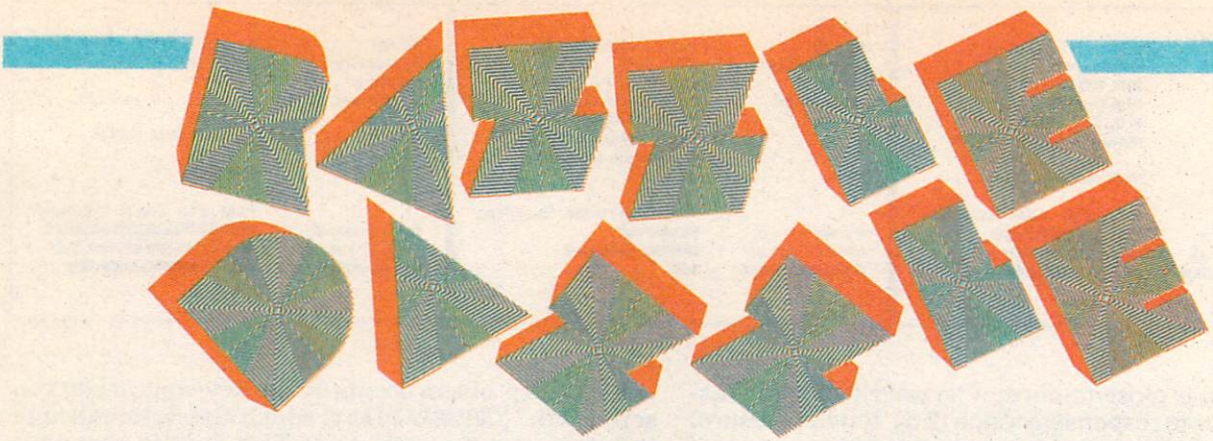
A spokesman for Keytronic has acknowledged this bug and assured us that if you send in the defective ROM chip to Keytronic, they will replace it—although the current supply of manufactured keyboards will not be recalled for correction.

Clockwise from left: Cherry, old IBM board, new IBM board, Keytronic.

"Are these two new third-party models superior enough to IBM's release to warrant the added cost of purchasing either one?"

Is It Worth It?

The major question facing a PCjr owner when contemplating either the Cherry or Keytronic keyboard, is whether the third-party keyboards are worth the extra $225 (Keytronic) or $235 (Cherry). If you are an occasional PCjr user and don't feel the need for a numeric key pad or single keystroke function keys, then $225-$235 does seem a bit steep. However, if you use the PCjr for word processing or work with programs that require a lot of numeric entry, or if you are involved in program development projects, the price may be well worth the added convenience—especially in the case of the Keytronic board. The Cherry keyboard just doesn't offer enough versatility or ease-of-use to warrant the added expense. I have used a variety of keyboards from Adler to Xerox, and the Keytronic ranks right up there with the best of them. HCM

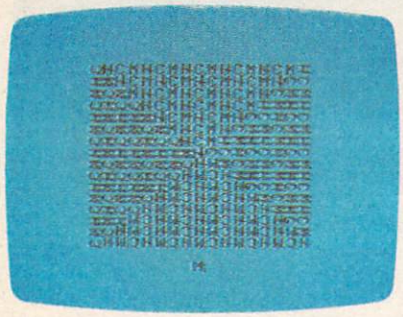


by William K. Balthrop
HCM Staff

Razzle 'n' dazzle 'em with spectacular video effects as you explore the mysteries of home computer graphics. Pixel magic is finally within your power . . .

With all the new technology on the market today, you may wonder whether the TI-99/4A can still compete as a powerful graphics computer. You bet it can! The TI is capable of a number of things that the newer, more expensive machines balk at.

Here, for example, we present one method of using the TI-99/4A as an effective graphics tool—in ways you may not have considered. This routine requires an Extended BASIC cartridge.



This photo shows how Turn can be used to create spectacular effects with very little programming effort.

Character Graphics

On the TI-99/4A, you can redefine the character set to create your own custom characters. This allows you to design spectacular graphics on the screen.

All of the characters you see on your screen are actually made up of very small dots called *pixels*. Each character has 64 pixels, laid out in an 8x8 grid. By telling the computer which pixels you want turned on (visible on the screen), you can change a character's shape. Each digit of hexadecimal code in a character's definition controls four pixels (dots within the character). It takes 16 hexadecimal digits to define the pattern for one character.

In Figure 1, pixels that are turned on are designated by a 1; those turned off are designated by a 0. The character "A" is formed by the *on* pixels.

Usually a programmer is left to figure out the hexadecimal codes after defining the new shapes with a piece of graph paper and pencil. But this program, *Characters*, contains three Extended BASIC subroutines which alter the character set automatically. The three routines here take the patterns for the characters passed to the routine, alter them, and then store the new shapes at the character location you specify.

FIGURE 1
CHARACTER BIT PATTERN

Row	Bit Pattern	HEXDigits
1	0 0 0 0 0 0 0 0	00
2	0 0 1 1 1 0 0 0	38
3	0 1 0 0 0 1 0 0	44
4	0 1 0 0 0 1 0 0	44
5	0 1 1 1 1 1 0 0	7C
6	0 1 0 0 0 1 0 0	44
7	0 1 0 0 0 1 0 0	44
8	0 1 0 0 0 1 0 0	44

Character definition string for the "A" graphics pattern = "003844447C444444"

Mirror

The first subroutine, Mirror, will cause characters to appear as their mirror images—turning them around as if you were looking at them from the other side of the screen. This routine takes the hexadecimal code for each row of pixels in the characters and reverses them.

Each character has eight pixels per row. Consider a row of pixels that has the following pattern:

11001010 CA

First we need to switch the location of each hexadecimal character to produce the following:

10101100 AC

The task now is to reverse the patterns within each four-pixel half of the row. This is done by reversing the bit pattern for each hexadecimal digit:

01010011 53

This process is then repeated for each of the other rows.

Flip

The second subroutine, Flip, will do just what the name implies: It will turn the characters it affects upside down. This process is easier to accomplish than the first routine—all that is needed is to reverse the order of the rows in the character's pattern definition. Remember, each row is made up of eight pixels, or two hexadecimal characters. The subroutine works on the character definition string as follows:

0123456789ABCDEF (Original hex codes)
01 23 45 67 89 AB CD EF (Row pairs)
EF CD AB 89 67 45 23 01 (Reverse order)
EFCDAB8967452301 (Final result—flipped)

". . . by redefining the character set to create custom characters, you'll be able to design spectacular graphics right on the screen."

Turn

This final subroutine, Turn, can be used to turn a character clockwise 90 degrees (on its side). If you repeatedly turn the same character, placing the modified shape back in that character's original location, you will continue to rotate it clockwise.

This routine is much more complex than the previous two, and consequently is much slower. Having explained Flip and Mirror, we'll leave it as an exercise for you to discover how Turn works.

Use Them Yourself

You can use these routines in your own programs to manipulate any characters available on the computer with Extended BASIC (ASCII 32 through 143).

All three subroutines require that you pass two parameters. The first parameter is a string containing the characters you want to convert. The second parameter is an ASCII value that indicates where the new characters will be placed. The following are examples:

CALL MIRROR("ABC",96) Place a mirror image of A in ASCII character 96, B at 97, and C at 98.

CALL FLIP("ABC",65) Turn the characters A, B, and C upside down and place the new shapes back into their original position, starting at ASCII 65.

CALL TURN("ABC",65) As used in the demo routine supplied with these subroutines, the letters ABC are turned 90 degrees clockwise with each call of the routine. In addition, before rotating the characters, we placed the shapes for the characters HCM at ASCII locations 65, 66, and 67.

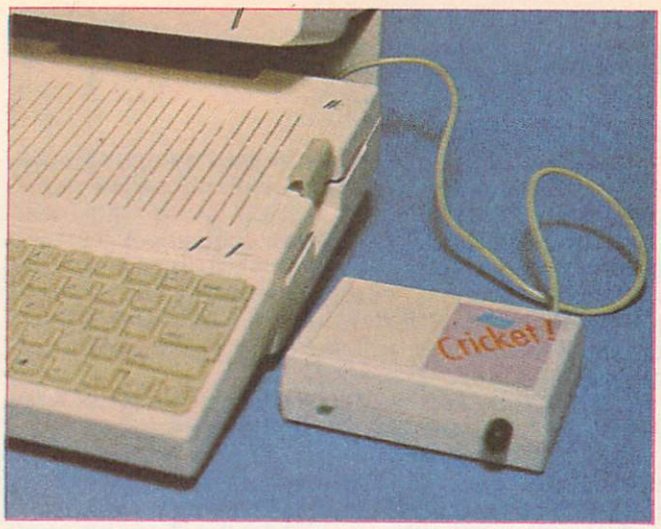
HCM

XB *Razzle Dazzle* requires TI Extended Basic.

Characters (TI-99/4A) Explanation of the Program

Line Nos.	
100-160	Program header.
170-330	Demo to access the three subroutines. Draws a spiral of the letters HCM.
340-400	Mirror subroutine.
410-460	Flip subroutine.
470-590	Turn subroutine.

```
100 REM  *********
110 REM  *  CHARACTERS  *
120 REM  *********
130 REM  BY WILLIAM K BALTHROP
140 REM  HOME COMPUTER MAGAZINE
150 REM  VERSION 4.5.1
160 REM  TI EXTENDED BASIC
170 CALL CLEAR
180 DISPLAY AT(12,1)ERASE ALL:"WORKING!"
190 N$="HCM" :: FOR Z=0 TO LEN(N$)-1 :: CALL CHARPAT(ASC(SEG$(N$,Z+1,1)),A$) :: CALL CHAR(65+Z,A$) :: NEXT Z
200 N$="ABC" :: CALL TURN(N$,65) :: FOR Z=0 TO LEN(N$)-1 :: CALL CHARPAT(65+Z,A$) :: CALL CHAR(96+LEN(N$)+Z,A$) :: NEXT Z
210 CALL TURN(N$,65) :: FOR Z=0 TO LEN(N$)-1 :: CALL CHARPAT(65+Z,A$) :: CALL CHAR(96+(2*LEN(N$))+Z,A$) :: NEXT Z
220 CALL TURN(N$,65) :: FOR Z=0 TO LEN(N$)-1 :: CALL CHARPAT(65+Z,A$) :: CALL CHAR(96+(3*LEN(N$))+Z,A$) :: NEXT Z
230 X=12 :: Y=16 :: P=0 :: FOR Z=2 TO 16 STEP 2 :: FOR Z1=1 TO Z-1 :: CALL HCHAR(X,Y,96+P) :: Y=Y+1 :: P=P+1 :: IF P=LEN(N$) THEN P=0
240 NEXT Z1
250 FOR Z2=1 TO Z-1 :: CALL HCHAR(X,Y,96+LEN(N$)+P) :: X=X+1 :: P=P+1 :: IF P=LEN(N$) THEN P=0
260 NEXT Z2
270 FOR Z3=1 TO Z :: CALL HCHAR(X,Y,96+(2*LEN(N$))+P) :: Y=Y-1 :: P=P+1 :: IF P=LEN(N$) THEN P=0
280 NEXT Z3
290 FOR Z4=1 TO Z :: CALL HCHAR(X,Y,96+(3*LEN(N$))+P) :: X=X-1 :: P=P+1 :: IF P=LEN(N$) THEN P=0
300 NEXT Z4 :: NEXT Z
310 CALL HCHAR(22,16,98) :: GOSUB 330 :: CALL HCHAR(22,16,101) :: GOSUB 330 :: CALL HCHAR(22,16,104) :: CALL HCHAR(22,16,107)
320 GOSUB 330 :: GOTO 310
330 FOR TD=1 TO 20 :: NEXT TD :: RETURN
340 REM  *********
350 SUB MIRROR(A$,A)
360 X$="0123456789ABCDEF" :: FOR X=1 TO LEN(A$) :: C$=SEG$(A$,X,1) :: FOR Z=1 TO LEN(B$) :: D$=D$&SEG$(B$,Z,1) :: NEXT Z :: CALL CHAR(A,D$) :: A=A+1 :: NEXT X :: SUBEND
370 SUB FLIP(A$,A)
380 FOR Z=1 TO LEN(D$) STEP 2 :: E$=SEG$(D$,Z+1,1)&SEG$(D$,Z,1) :: CALL CHAR(A,E$) :: A=A+1 :: IF A>143 THEN SUBEXIT
390 NEXT X
400 SUBEND
410 REM  *********
420 SUB FLIP(A$,A)
430 FOR X=1 TO LEN(A$) :: CALL CHARPAT(A$,Z,2) :: E$=E$&SEG$(B$,Z,2) :: NEXT Z :: CALL CHAR(A,E$) :: A=A+1 :: IF A>143 THEN SUBEXIT
440 E$="" :: FOR Z=LEN(B$)-1 TO 1 STEP -2 :: E$=E$&SEG$(B$,Z,2) :: NEXT Z :: CALL CHAR(A,E$) :: A=A+1 :: IF A>143 THEN SUBEXIT
450 NEXT X
460 SUBEND
470 REM  *********
480 SUB TURN(A$,A)
490 FOR Z=0 TO LEN(A$)-1 :: CALL CHARPAT(A$,Z+1,1),B$) :: FOR Y=1 TO 8 :: P(Y)=0 :: NP(Y)=0 :: NEXT Y
500 FOR Y=1 TO 15 STEP 2 :: B1=ASC(SEG$(B$,Y,1))-48 :: IF B1>9 THEN B1=B1-7
510 B2=ASC(SEG$(B$,Y+1,1))-48 :: IF B2>9 THEN B2=B2-7
520 P(INT(Y/2)+1)=B1*16+B2 :: NEXT Y
530 FOR Y=1 TO 8 :: FOR X=0 TO 7 :: IF (P(Y)AND 2^X)=2^X THEN NP(8-X)=NP(8-X)+2^(Y-1)
540 NEXT X :: NEXT Y
550 NP$="" :: FOR Y=1 TO 8 :: B1=INT(NP(Y)/16) :: B2=NP(Y)-B1*16 :: IF B1>9 THEN B1=B1+55 ELSE B1=B1+48
560 IF B2>9 THEN B2=B2+55 ELSE B2=B2+48
570 NP$=NP$&CHR$(B1)&CHR$(B2) :: NEXT Y
580 NEXT Z
590 SUBEND
```



The Cricket!

—A Sound Synthesizer And More For The Apple

A review by Dana M. Campbell
HCM Staff

All alone with no one but your Apple to talk to? Just give a little whistle for the Cricket!

There's a new gadget on the market that will make your Apple computer talk, chirp, chime, rhyme, and ding-a-ling in any BASIC program you ever write—even in stereo. What is it? It's called a *Cricket!*, but it's nothing like the insect variety.

The *Cricket!* speech and sound synthesizer is a small plastic box that resembles those old pocket-sized transistor radios. It connects to the Apple *IIc*'s modem port, or to a serial card in slot 2 of the Apple *IIe*. (A custom cable is required to hook it up to an Apple Super Serial card in the *IIe*, and the cable configuration for this is shown in Appendix D of the manual.) The volume may be adjusted with a side knob, and there is a stereo output jack on the back of the *Cricket!* to connect speakers or a headphone. A power transformer and program disk are included in the package.

Once you've plugged in the *Cricket!* and booted up the disk, it's a simple matter of selecting EXIT TO BASIC from the Main Menu to program it into your own BASIC routines. The *Cricket!* program occupies the Apple's "extended" memory so that all of the normal BASIC programming memory is available for use.

". . . make your Apple computer talk, chirp, chime, rhyme, and ding-a-ling in any BASIC program you ever write—even in stereo!"

Can It Really Talk?

Four basic features are available in the *Cricket!* for use with your programs: speech, music, sound effects, and a clock. The speech alternative has three options, differing in ease of use and controllable variables—which in turn affect the speech quality.

The Unlimited Vocabulary Speech TALK command system is the easiest option to use and is somewhat flexible. A robotic male voice, guided by more than 400 pronunciation rules, takes into account speed, pitch, inflection, and volume in pronouncing English text. In addition to the above variables, the TALK commands prepare the *Cricket!* to spell words out letter by letter; pronounce all punctuation characters (especially handy for % and $); and speak all text printed to the screen, thus eliminating the need for TALK commands to directly generate speech (at least until you want to change your parameters again).

Of course, anyone who has ever struggled with the intricacies of proper grammar knows that there are always exceptions to the rules, and these exceptions are sometimes evident here. For the *Cricket!* to master all the nuances of pronunciation, it is sometimes necessary to use "creative spelling" (spelling out the words in your string according to how they sound), or to simply break words up into syllables.

You Don't Say . . .

The Fixed Vocabulary Speech SAY command system is more limited in its vocabulary range and flexibility than the TALK system, yet it offers the most realistic speech available—a natural sounding female voice. Before you can start punching in commands and start hearing the lifelike speech, however, you must first compile the desired words for your message using the Word Editor option. Once you have chosen the words you want from the 725 word list, it is a simple matter to save your list, load it, and start hearing speech. You can add prefixes and suffixes to words to expand your list.

The Unlimited Vocabulary Phonemic Speech PHN command system, like the TALK system, offers an unlimited vocabulary and a male robotic voice. However, the TALK system uses those 400 pronunciation rules to first analyze a word and then speak it. But with the PHN command, the *user* must define how the words are to be pronounced by using "phoneme codes," as well as inflection, stress, pitch, volume, and pause symbols.

Because there are so many variables that can be manipulated by the user when in this mode, it may at first seem complicated to use. It is not. True, there is a lot to remember in creating each word, but with the help of the manual's phoneme code index (and perhaps a dictionary), there is no chance you'll get lost—and the flexibility in word creation offered by phonemes is worth the effort. It will probably take quite a bit of experimentation to learn how to achieve the desired sound of your words; but at the same time you can create entirely new languages by playing with these variables, and have a great time of it as well.

Hit It Maestro!

Although a music editor program (available separately from the package) is required to take full advantage of the music capabilities of the *Cricket!*, the device alone is still able to play music using up to six different simultaneous voices or melodies. From BASIC, you can play a one-note melody with a range of up to eight octaves, accompanied by three note chords. You can use a TEMPO command to adjust the speed at which your songs are played, varying from settings of 1 to 255. The DECAY command adjusts the length of time it takes for a note to fade away. People knowledgeable about music will find these and the other music-writing tools to be helpful, with a lot of room for creativity. Those less knowledgeable will begin learning something about composing music, and can still create some wild tunes to accompany their game programs, for instance.

A nice feature about the *Cricket!* is that it has two separate sound generators, so two different sound effects can be played at the same time. Again, before you can hear anything, you must first go into an editor and create your sound. The Sound Editor screen is overwhelming if you haven't read the manual and are unfamilar with the elements of sound, as I was. However, I found that observing and modifying the settings of the sample sounds, like Gallop, Ocean, and Gunshot, gave me some idea of the relationship of each of these elements to each other, and of their individual effect (especially the envelope choices). The next step is to start creating your own effects and saving them to disk for a sound library.

"It inherently serves as a sort of learning program, making you aware of the structure and components of speech and music, the elements of sound, and even spelling . . ."

The Clock function on the *Cricket!* allows you to use time, date, and even alarm functions with your system, and, because the *Cricket!* has its own power source, will work whether your computer is on or off. (Unfortunately, you cannot create the actual alarm sound—it simply rings six chimes.) The *Cricket!* manual describes how to modify ProDOS to save the time and date with all ProDOS files, much like the Thunderclock clock card mentioned in the ProDOS manual.

Only Average Documentation

Although the manual for the *Cricket!* is not poor, it's not outstanding either—simply adequate. All of the commands and information you need to add sound to your programs are included in a concise, clear style, and it is well-indexed. Each major mode has a separate section explaining its operation, and the appendices are most helpful. Advanced programming information is provided so that you can ignore the *Cricket!* program when using the clock or music functions, or to use it from machine language.

However, the writers went overboard in being concise; if the manual included information on *why* each sound element creates and affects noise, some of the definitions would be more meaningful, and one could spend less time experimenting just to achieve a desired sound. For those of us who can write a complicated applications program but who are not exactly sure what effects tone and pitch have on sound, and what relationship these various sound elements have to each other, even a short little explanation would be an immeasurable boon.

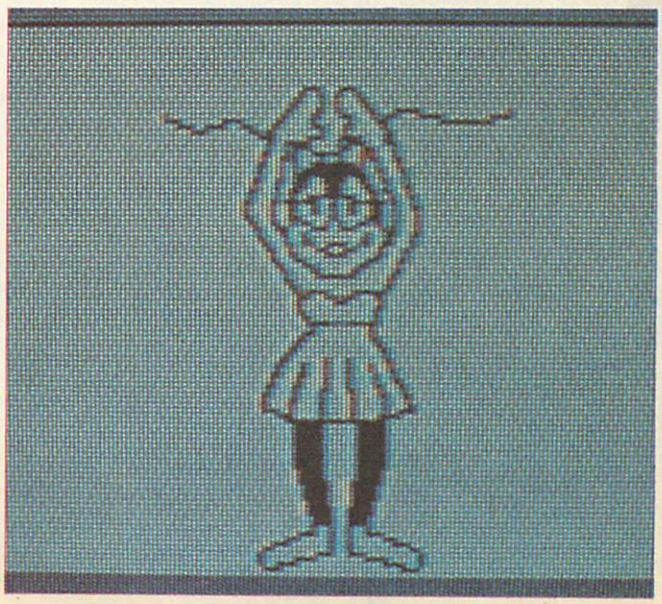
Name:	The Cricket!
Product Type:	Speech and Sound Synthesizer
Machines:	Apple *//e*, *//c*
Distributor:	Street Electronics Corp. 1140 Mark Avenue Carpinteria, CA. 93013 (805) 684-4593
Price:	$179
System Requirements:	*//c*: ProDOS *//e*: ProDOS, extended memory 80-column card, Street Electronics' Alphabits card or Apple Super Serial card.

	Poor	Fair	Good	Excellent
Performance:				■
Ease of Use:			■	
Ease of Set-up:				■
Documentation:		■		

A Good Value

I am disappointed that the explanations are kept to such a minimum, because the rest of the package has so much to offer. Many synthesizer packages on the market are much more limited than the *Cricket!* in their offerings, often providing only one or two of the above features. Of course, they don't cost as much as the *Cricket!*, but by the time you bought three or four synthesizers to give you sound, and music, and speech, and a clock, your expenditure would far exceed the price of a *Cricket!* In addition, the *Cricket!* inherently serves as a sort of learning program, making you aware of the structure and components of speech and music, the elements of sound, and even spelling (the Spelling Test option allows you to choose words with the Word Editor to practice your spelling). The learning process would have been complete if, as mentioned above, we had more whys and wherefores to ponder.

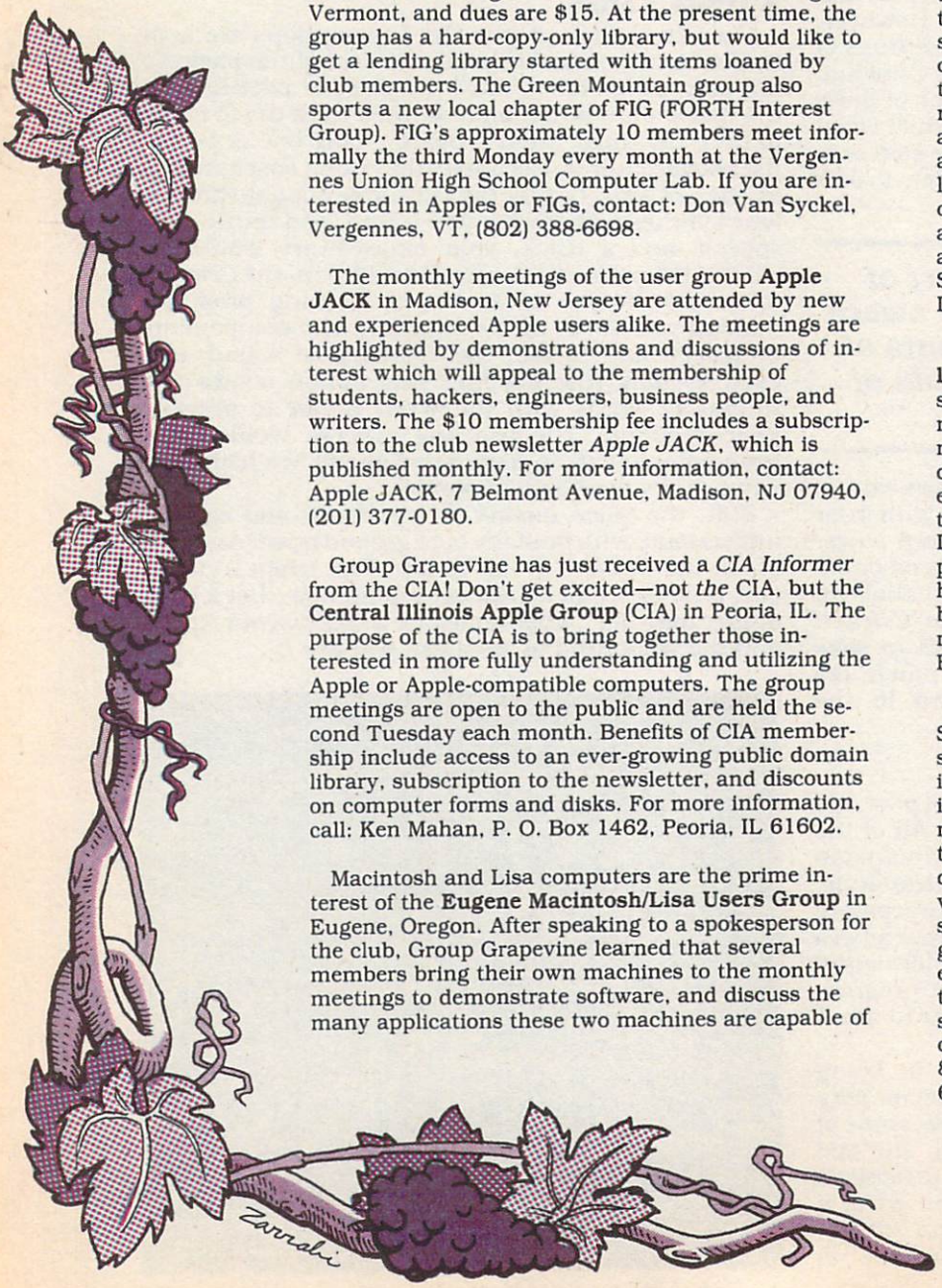
Still, the voice quality is wonderful and easy to understand, with no static background noise. And best of all, the *Cricket!* is fun to use. Tip: when trying it out, type in a quick TALK command like "Let's have dinner together" when someone walks by your Apple, and see what kind of response you get . . . **HCM**



Group Grapevine

News, information and upcoming events of home computer users groups around the world.

Looking to join a users group, exchange newsletters or software, increase your users group's membership or pep up your next meeting's agenda? For the latest users group news, put your ear to the Group Grapevine. And if you have a message to put out to other groups, if you are starting a new group, or have an interesting item to share, send a note or picture—or better yet, a group newsletter—to the Users Group Editor, Home Computer Magazine, 1500 Valley River Drive, Suite 250, Eugene, OR 97401, (503) 485-8796.



We've all heard of apples and oranges. Well, now we have Apples and FIGs! The **Green Mountain Apple Club** in Vergennes, Vermont meets the second Wednesday of each month at 6:30 p.m. in the Laureate Learning Center at Chace Mills, Burlington, Vermont, and dues are $15. At the present time, the group has a hard-copy-only library, but would like to get a lending library started with items loaned by club members. The Green Mountain group also sports a new local chapter of FIG (FORTH Interest Group). FIG's approximately 10 members meet informally the third Monday every month at the Vergennes Union High School Computer Lab. If you are interested in Apples or FIGs, contact: Don Van Syckel, Vergennes, VT, (802) 388-6698.

The monthly meetings of the user group **Apple JACK** in Madison, New Jersey are attended by new and experienced Apple users alike. The meetings are highlighted by demonstrations and discussions of interest which will appeal to the membership of students, hackers, engineers, business people, and writers. The $10 membership fee includes a subscription to the club newsletter *Apple JACK*, which is published monthly. For more information, contact: Apple JACK, 7 Belmont Avenue, Madison, NJ 07940, (201) 377-0180.

Group Grapevine has just received a *CIA Informer* from the CIA! Don't get excited—not *the* CIA, but the **Central Illinois Apple Group** (CIA) in Peoria, IL. The purpose of the CIA is to bring together those interested in more fully understanding and utilizing the Apple or Apple-compatible computers. The group meetings are open to the public and are held the second Tuesday each month. Benefits of CIA membership include access to an ever-growing public domain library, subscription to the newsletter, and discounts on computer forms and disks. For more information, call: Ken Mahan, P. O. Box 1462, Peoria, IL 61602.

Macintosh and Lisa computers are the prime interest of the **Eugene Macintosh/Lisa Users Group** in Eugene, Oregon. After speaking to a spokesperson for the club, Group Grapevine learned that several members bring their own machines to the monthly meetings to demonstrate software, and discuss the many applications these two machines are capable of performing. At this writing, the group is looking for a permanent home, so if you are interested in becoming a member or are a Macintosh or Lisa group outside the Eugene area, drop a line or call: P. O. Box 10988, Eugene, OR 97440, (503) 345-2393.

Thoroughbred horses aren't the only good things you'll find in the bluegrass country of Kentucky. You'll also find the **Bluegrass IBM PC User Group** in Lexington, Kentucky, meeting the fourth Saturday of every month at the University of Kentucky's McVey Hall, Room 231. Membership consists of about 130 non-technical people who enjoy getting together and talking about their computer experiences and expertise. Popular among the members is the open forum segment of the meeting, when ideas and experiences can be shared in an informal atmosphere. Special Interest Groups (SIGs) are beginning to form for beginners and C language fans. An assembler SIG is already in existence. Public domain software is available on approximately 30 disks in the club library and members are encouraged to make copies of the software for their own use at no charge. If you are an IBM PC owner living in the bluegrass country and would like to join an active group, contact: Diane Skoll, Room 72, McVey Hall, University of Kentucky, Lexington, KY 40502-0045, (606) 257-2900.

Remember back in 1982 when the **Northern Illinois IBM User Group** was formed? Well, they are still recognized as an IBM group, but with a new name—**Chicago Computer Society** (CCS). CCS's monthly publication, *PORT >fol >I/O*, contains product reviews, program listings, articles on current events, and listings of local happenings. In addition to their main meetings, members can attend Special Interest Group meetings which include C language programming, and local-area networking. CCS also has a 24-hour bulletin board system, making it possible for members to talk to all other chapters. For more information, contact: CCS, P. O. Box 95625, Hoffman Estates, IL 60195, (312) 642-6130.

The **Buffalo IBM PC User Group** already has Special Interest Groups (SIGs) in business & spreadsheets, data base and word processing—with a lot of interest in forming SIGs for assembly language and investments. A unique aspect of this group's meetings is what they call a "side show." Two or three vendors set up booths where members can see demonstrations and get hands-on experience with various equipment and software. Don says they are seeing more PCjr people becoming interested in the group and feels it's possibly due to the PCjr's new enhancements. The club meets the second Monday of the month at 7:30 p.m. at the Airway Hotel. Membership fee is $20, and just $10 for students and senior citizens. For more information, contact: Don Leslie, 84 Brookdale Drive, Williamsville, NY 14221 (716) 632-5748.

The **Pacific Northwest IBM PC Users Group** in Bellevue, Washington gets together the second Tuesday of each month to exchange information and ideas. This group formed in the summer of 1981 and has a staggering 550 members! The diverse interests of the members make this group quite interesting; their Special Interest Groups include Lotus 1-2-3, DOS, C language, generic financial model, communications, and BASIC. If you would like to learn more about this large group, call: Larry Shaw, P. O. Box 3363, Bellevue, WA 98004, (206) 628-1141.

Group Grapevine is pleased to announce the formation of the **Queensborough Community College 99'er User Group** in Bayside, NY. According to Frank Cotty, the group will begin meeting this fall and in keeping with the spirit of the community college, the group will be open to students and non-students alike. For more information, contact: Frank Cotty, 56 Avenue & Springfield Boulevard, Bayside, NY 11364.

Salute! Group Grapevine has received a letter from the 50-member **TI 99 IT User Group** in Montecchio, Italy. The group has a library of over 400 TI BASIC programs, more than 400 Extended BASIC programs, and a good quantity (and quality) of assembler programs. Their bimonthly bulletin includes free listings, tricks, tips, and descriptions of new programs. TI 99 IT is in contact with clubs in Belgium and Australia and would like to expand their contacts to include the U.S. If you would like to exchange newsletters, software, or program listings, write (in English): Perlini Settimio, TI 99 IT User Group, Via 21 Gennaio 152, 61020 Montecchio (PS), ITALY.

Chip Ragsdale, president of the **MIT Lincoln Laboratory TI-99/4A User Group** has informed Group Grapevine that they are no longer in existence (in Massachusetts) due to his transfer to Florida. Since Chip has been in Florida, he has organized a new MIT group and they are 52 members strong at the present. Their group meets once a month, and they are presently working on their first newsletter. They also have a cassette library which boasts 480 programs, with many more yet to be keyed in. For more information, contact: Chip Ragsdale, 8820 90 Way North, Seminole, FL 33543.

Larry Robinson, vice president of the **Macon Area TI-99/4A User Group**, has informed Group Grapevine of the formation of the **Warner Robins Chapter**, which meets at the Warner Robins Recreation Center from 10 a.m. to 1 p.m. every Saturday. A lecture series on BASIC programming techniques and hardware use has been continuing for some time. The group also has tentative plans for hosting a computer exposition featuring displays by area users' groups. This is designed to spread the word about users' groups in the middle Georgia area. If you are interested in more information about the Macon Area group or the Warner Robins Chapter meetings, write: Larry Robinson, 503 Third Avenue, Bonaire, GA 31005.

Here's a group that has come a long way in a very short time. They are the **Crossroads Commodore Users Group** in Victoria, Texas. They have been in existence for about eight months and boast a membership of 45. According to spokesperson Jerry Guy, they were originally a support group for the Commodore users of Victoria and vicinity who are out there in the "wasteland" between Houston, San Antonio, Austin, and Corpus Christi. (Group Grapevine looked up Victoria in an atlas and they really *are* stuck out there—if you draw lines between the cities mentioned above, it *is* a "crossroads.") The original members of the group began swapping programs, which has resulted in a library of about 500 programs. They are affiliated with the Commodore Houston Users Group (HUG) and the Toronto PET users. Membership is $3.00/month. If you are interested and would like more information, contact: Jerry Guy, 417 Irma Drive, Victoria, TX 77901, (512) 575-0342.

Bloodthirsty! That's how Mr. Ikram of a Commodore users group in **Rochdale, England** describes his group with reference to exchanging good American software for English software. The group prefers cassette software unless American groups or individuals can duplicate some of their disk or cartridge software to tape. The group would also enjoy receiving newsletters and any other information from American Commodore groups. If you are interested in contacting this group, write to: M. Ikram, 20 Equitable Street, Rochdale OL11 1JQ, Lancashire, England, UK.

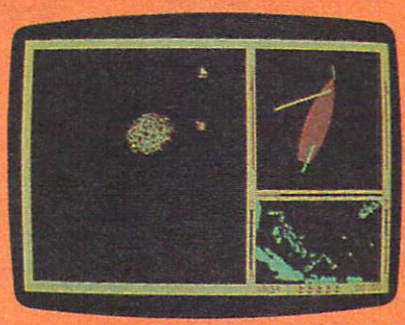
We received greetings from the **Lansing Area Commodore Club** in East Lansing, Michigan via president Jae Walker. LACC has been in existence for a year-and-a-half and is a whopping 240 members strong. They meet on the second Thursday of each month at All Saints Episcopal Church at 7 p.m. Anyone wishing further information can contact Jae at: P. O. Box 1065, East Lansing, MI 48823-1065, (517) 351-7061.

Group Grapevine just received notice of the birth of another Commodore 64 users group. The **Akron Area C-64 Users Group** in Cuyahoga Falls, Ohio has a membership of well over 100, with interests ranging from business and education to gameplaying, ham radio, etc. They meet the fourth Saturday of each month from 1 to 4 p.m. at the Green Middle School, Green Township, just south of Akron. The Green Middle School has a computer lab with 15 Commodore 64's that are available for use by the membership. If you are a Commodore 64 owner and would like more information about this users group, write or call: Paul M. Hardy, 2453 Second Street, Cuyahoga Falls, OH 44221, (216) 923-4396. **HCM**

Sailing

A review by
Steve Nelson
HCM Staff

HCM Review



Name:	Sailing
Program Type:	Adventure game
Machines;	IBM PC, PCjr
Distributor:	Accupipe Corporation 222 West Lancaster Ave. Paoli, PA 19301
Price:	$34.95
System Requirements:	IBM PC: color graphics board, 128K. IBM PCjr: 128K.

	Poor	Fair	Good	Excellent
Performance:				
Engrossment:				
Documentation:				

***Are you up to the adventure of sailing the treacherous waters of the Bermuda Triangle?* Beware the dangers, Mate . . .**

For most people, life is interesting enough without risking life and limb in pursuit of buried treasure or some other such will-o'-the-wisp. But I think that in each one of us, there burns the urge to test ourselves against the unknown. In some people, it is just a tiny spark, while in others it smolders—and in a very few it is an all-consuming fire that drives them to wander the world seeking adventure.

There are plenty of possible adventures to choose from: one could go to Nepal and trudge through hip-deep snow, risking frostbite and hypothermia to search for the Abominable Snowman, or perhaps look for a lost Spanish gold mine in the Superstition Mountains of Arizona in 120 degree heat. Is this too much adventure for you?

Beware The Dangers...

Well then, here's an alternative. In Accupipe's adventure game, *Sailing*, you can sail on a journey across the screen of your IBM PC or PCjr while attempting to rescue swimmers and rafters adrift inside the infamous "Bermuda Triangle." A bonafide adventure if ever there was one. "Sounds easy," you say? If you can maneuver your boat through alien crystals, demonic storms, gravitational vortexes, speed boats, sharks, and the dreaded creeping mist, then you are well on your way to becoming a world-class adventurer.

You begin *Sailing* on the first of three legs of your journey, setting sail for Bermuda from Puerto Rico. The object of the game is to sail all three legs of the Triangle, picking up stranded swimmers and rafters for points while racing the clock back to port.

You control the boat by manipulating the sail with the keyboard, allowing it to catch the wind (which always blows South to North), and by moving the rudder to turn to port or starboard. The boat is quite responsive to keyboard input, and once you get your "sea legs" you will have no trouble staying on course.

You must, however, be careful not to inadvertently press the wrong key. This can be a problem when changing direction—causing you to stall, or simply travel in the wrong direction at the wrong time. Until you become familiar with the controls, you may find yourself trying to turn away from a hazard and end up sailing right for it.

A 3-Window Screen

Once under way, you can sail the boat in any direction and observe your progress on the monitor. The screen is divided into three "windows": (1) a close-up of the sailboat itself, showing sail, rudder position, and the *telltale* (which indicates the direction of the wind); (2) a radar screen showing the entire course; and (3) a medium-range view showing the position of the sailboat in relation to objects nearby.

Sailing's graphics are excellent, using some of the most brilliant colors I have seen on a monitor. A fine example of the IBM machines' graphics capabilities is the creeping mist, which looks like a swarm of multicolored bees hovering above the rich, blue ocean. (You can use a color television with the PCjr, but the screen resolution is not as crisp as that on an RGB monitor.)

Because *Sailing*'s graphics are so spectacular, I expected the sound effects to be almost as good. Unfortunately, the variety of sounds is somewhat limited. However, as sparse as they are, the sound effects are effective, and generally add to the enjoyment of the game. (You have the option to turn off the sound if you wish.)

Meet The Challenge, Mate

Overall, *Sailing* is quite challenging, with the other two legs of the voyage (Bermuda to Miami, and Miami to Puerto Rico) becoming progressively more difficult—you will be sailing against the wind. You must learn how to sail into the wind using a series of "tacks." Old Salts will have no trouble doing this, but if you're a swabbie (as I was), tacking can be difficult to master—there is always the possibility of luffing (stalling caused by heading into the wind). Luckily, the boat comes with an engine, but be warned—there is a limited amount of fuel; once it is gone, you will be helpless.

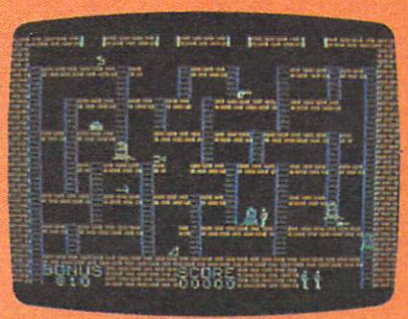
Game instructions are short, and give you all the information you need to maneuver your boat through the Bermuda Triangle (at least on a screen). Most of the instructions are given on-screen, with effective graphics and clear text. It would be nice, however, if this information was also in print, so you could refer to it while playing. (There is one slight error in the screen instructions: you are told to expect leg time on the right side of your control panel, and total score on the left, but the reverse is true). Unfortunately, the instructions don't tell you how to avoid all the dangers. But if there wasn't any unexpected danger, it wouldn't be much of an adventure, would it?

Accupipe seems to have a winner in *Sailing*. It is realistic, imaginative, and while it may not be the same as actually sailing the Bermuda Triangle, it may help cool that fire burning in you. And after all, isn't that what adventures—real or imagined—are all about? HCM

Midnite Mason

A review by
Steve Nelson
HCM Staff

HCM Review ✧ Diamond-In-The-Rough



Name:	Midnite Mason
Program Type:	Arcade game
Machine:	TI-99/4A
Distributor:	Software Specialties, Inc. P.O. Box 3304 Evergreen, CO 80439
Price:	$24.95 Cartridge
System Requirements:	Joysticks optional

	Poor	Fair	Good	Excellent
Performance:				
Engrossment:				
Documentation:				

A mason's work is never done—especially when the building he is working in is haunted by four nocturnal nasties.

Midnite Mason is an entertaining little game that pits your speed and strategy against a group of pesky ghosts. It's four against one, and if you take a wrong turn, you lose. The scenario is this: You are a mason worker who has left his tools inside a building. No problem, just go back and get them, right? Wrong! At night, the building is haunted by four nasty ghosts who seem to like your tools where they are. What's worse, all of your tools are scattered around the building and the only way to get them is to climb ladders to several different floors, dodge the ghosts, and grab your tools and run. To recover a tool, just run past it, but watch out for those spooks—one touch and you're a goner!

When you successfully recover all 7 tools, the ghosts vaporize, and you advance to the next level of play. *Midnite Mason* has 6 different mazes (buildings), and 4 levels of difficulty—the ghosts actually get more intelligent as they pursue the mason. Once you successfully complete the fourth level, the ghosts' intelligence decreases, but their speed increases. Got all that? In other words, the ghosts get smarter, then dumber and faster. The game continues like this indefinitely.

Making the game even more difficult is a time limit for gathering up your tools. A timer counts backwards from 900 to 000 in increments of 10, forcing you to keep moving. If you run out of time, you lose one of the three masons.

The response of the game is very good, except that the mason sometimes hesitates at the top of ladders before moving left or right, and as a result, usually gets mugged by a passing ghost. I didn't have this problem when using the keyboard, which makes me suspect the joystick-read routine. I tried out several different joysticks and the problem was still there. (It doesn't happen very often but it can cost you a mason when it does.) The mason can elude the ghosts by running past ladders, because the ghosts tend to climb them instead, even in the heat of the chase.

Pick A Hole

The mason, however, is not totally defenseless. He can chop a hole in the maze with his pick very quickly and cause a ghost to fall to the lower level. He can also fill in a hole amazingly fast, allowing him to escape from one tier to another, then quickly turn around and chop a hole again, forcing any pursuing ghosts to find another way to get to him, and giving him more time to gather his tools.

Although *Midnite Mason's* game plan sounds complicated, it is very simple to play. The mason's speed is agonizingly slow, especially when a couple of the ghosts are hot on his trail; but with strategy, you can avoid the touch of death—at least for a while.

I found *Midnite Mason* to be an enjoyable game, and one that is difficult enough to require a determined effort on the part of the player in order to score points. Once you earn 5000 points, you receive a new mason—and by then you will probably need one.

The documentation is brief, but thorough, except that it fails to tell you to release the (ALPHA LOCK) key when using joysticks.

Midnite Mason is a good game, but it could be better. For instance, as you get past the first four levels, the number of tools that the mason must retrieve should increase. Also, the mason should be able to jump off the ladders if the ghosts trap him. And finally, it would be nice if the program could keep track of the high score while the cartridge is engaged, giving you something to shoot for on each repeat game you play. Incorporating these changes would make *Midnite Mason* one of the better games of its type.

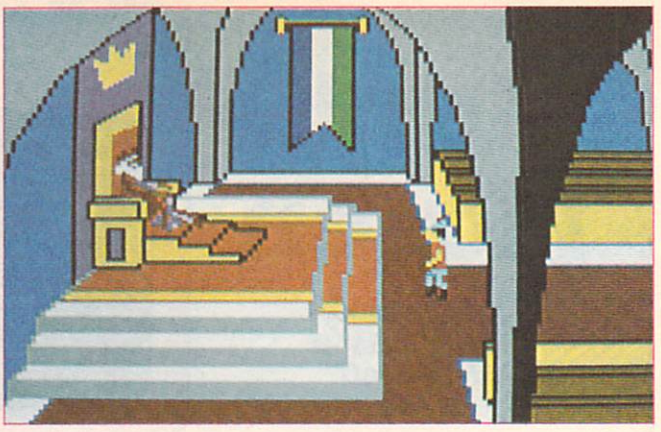
"The mason's speed is agonizingly slow, especially when a couple of the ghosts are hot on his trail; but with strategy, you can avoid the touch of death—at least for a while."

Video game players who are used to complex game plans and hordes of monsters or aliens to kill will probably find *Midnite Mason* a bit on the slow side. Compared to *Buck Rogers and the Planet of Zoom* (Reviewed in Vol. 4, No. 4 of *Home Computer Magazine*), its level of excitement and story line are somewhat mundane, but I found its relative slowness and simplicity to be refreshing, and a lot easier on the eyes.

The game's sound effects are humorous and well done, and I love the little dance the mason does when he gets all 7 tools and the ghosts vaporize.

Overall, this is a very good version of a well-used story line—get prizes in the maze, watch out for big meanies. The screen graphics are excellent, however, except for a few minor complaints, *Midnite Mason* is a winner.

HCM



King's Quest

A review
by William K. Balthrop
HCM Staff

In the beginning, there were simple text adventures. Then came text adventures with static background graphics. And now—full-color 3-D interactive-graphic adventures with sound effects! Where will it all end?

In all my years of working with computers and games, I have often dreamed of the ultimate computer adventure—one where you can actually see yourself moving about the surroundings, hear the birds and animals, and have complete control over your physical movements. I am very pleased to let you know that this dream came true one afternoon when *King's Quest* was put on my desk. I have played both computer and tabletop (role-playing) adventure games, but have never seen anything that parallels this new graphics adventure.

King's Quest mixes state-of-the-art graphics and animation with the best qualities of the text adventure. Your screen fills with a three-dimensional picture of a castle, complete with moat and bridge. There are trees, and grass, and mountains on the horizon—no important detail is spared. In the opening scene, your character, Sir Grahame, stands alongside the moat on the far side of the bridge from the castle. His mission: to find and retrieve a jewel-inlaid treasure chest, an enchanted mirror, and a magic shield.

Three Dimensions

Now, at first glance, you might suspect that this is just another adventure game with good graphics. If so, you're in for a surprise. Try moving Sir Grahame using your joystick or the keyboard arrow keys. Notice the smooth, realistic animation exhibited as Sir Grahame walks, not floats, around the screen.

Oops! Bumped into a tree. That's okay, just walk behind it. What? Sir Grahame *disappears behind the tree!* What magic is this? You bring Sir Grahame back out from behind the tree and go around the other way. You think, "This isn't happening," and reach for your glasses as Sir Grahame now walks in *front* of the tree. You touch the screen to see if it's really become three-dimensional. Has Sierra On-Line (developers of this program for IBM) hired a group of wizards to do their programming?

Technically speaking, I have a few theories as to how they did it. However, trying to explain complex animation and 3-D graphics techniques here would confuse the issue. Let's just say, "they did a great job."

Explanatory text is displayed at the bottom of the screen, where you can enter explicit instructions from the keyboard. For instance, when you find treasures and clues, you can take these items and examine them, open them, or give them away to someone else. Someone else? Yes, you're not alone in your adventure—you will encounter many friends and foes along the trail. How you should interact with each of these characters is something you will have to discover for yourself.

The world you will explore is called the Kingdom of Daventry. Daventry is full of many secrets, hidden treasure, and quite a few surprises. As you travel through the kingdom, you can move Sir Grahame from one area to the next by simply moving him off the edge of the screen. He will then appear on the opposite side of the next area (screen). Each screen is an adventure in itself, with a new set of items and clues you will need to explore in order to unlock some of the game's secrets. I was able to find at least 69 separate areas (screens) around the kingdom—sometimes at great risk to Sir Grahame. I'm sure there are more to find, because after all this traveling I had accrued only 42 of the game's possible 158 points.

The Kingdom of Daventry is designed like a globe. If you continue traveling in one direction long enough, you come back to the same spot you started from. It's like having your own miniature world.

Group Action

An important aspect of this game is the way it tends to draw groups of players into the action. It is not uncommon to see three or four people trying to be involved in the game at one time. You may even decide to assign separate responsibilities to each of several

Name:	King's Quest
Program Type:	Adventure Game
Author's Name:	Sierra On-Line
Machine:	IBM PCjr
Distributor:	IBM Corp. P. O. Box 1328-S Boca Raton, FL 33432
Price:	$50
System Requirements:	128K memory, IBM Color Display Monitor or a color television.
Optional Equipment:	IBM-compatible joysticks, external speaker if not using television.

	Poor	Fair	Good	Excellent
Performance:			■	
Engrossment:				■
Documentation:			■	

players: One person could control the joystick, while another enters commands at the keyboard, and yet another makes a detailed map of your progress through the kingdom. You could even have someone keep track of the clues, treasure, and miscellaneous items you encounter. This concept is a breakthrough for home computing in the area of adventure gaming. Before *King's Quest*, the adventure game focused on the single player—now the whole family can participate.

Key Features

The diskette that comes with this package is write- and copy-protected. However, the program allows you to copy the playing screens onto what's called the Play Disk. This can be done at any time during the game by entering COPY DISK on the command line. Once you have created a Copy Disk, you can save your games at any point and then recall them later to continue. This is a very important advantage, especially for beginners. Just before directing your knight to risk his life—save the game. Then, if Sir Grahame gets killed, you can recall the game and try another tactic. Three keys on the keyboard let you save a game, recall a previous game, or start a new game—each with a single key press.

"Never before has an adventure game involved so much detail and animation."

Sir Grahame can perform several actions with a single keystroke or the press of a firebutton. A colorful overlay provided for the old "chiclet" keyboard (which you can hang on to when you get your new keyboard) comes with the game, and indicates special functions and key commands for moving our fair knight around. You can "duck" out of danger, or "jump" into the air. There is also a lot of water in the Kingdom of Daventry—lakes and streams. Some of these you can safely enter, though you will have to press the (SWIM) key if you want to stay afloat. Once you start swimming, you can control your direction in the same way as if you were on land. But be careful; not all waters are safe.

A special key called (ECHO) lets you repeat a previously-entered command and redisplay it on the command line.

Numerous sound effects help add realism, such as rushing water when a waterfall is nearby, or the tweeting of birds, and the baying of an old billy goat. However, if you tire of the sound effects, you can turn them off and on with the press of a key.

After many hours of exploring, you may want to take a break to eat dinner (or breakfast, if you really get involved). You can temporarily suspend play, or restart the game with another key.

A (STATUS) key lets you know how well you're doing. It causes the screen to switch to a text display showing an inventory of items you have collected, and your current score.

Documentation

King's Quest comes with a 26-page manual which covers loading and running the program, and creating a Play Disk; it also provides you with enough information to start your adventure. The documentation is well-written and easy to understand. It doesn't go into great detail about the kingdom and its layout, or the characters you will meet. You will have to explore the kingdom for yourself. After all, half the fun is in figuring out which strategies work and which don't.

New Standard

Every once-in-awhile, a new software product comes along and sets a standard for all similar software to follow. *King's Quest* is that type of product. Never before has an adventure game involved so much detail and animation. It combines the best from arcade and text games, and adds a few things never before done on a home computer—creating one of the most exciting adventure games on the market. The three-dimensional realism is so fantastic that you'll be marveling at it for many hours.

This game provides *so many* puzzles to figure out that you probably won't successfully complete the adventure in the first week. There are many ways of accomplishing any of the several quests—the more difficult solutions are worth the most points. After completing a given quest, you can attempt to find a better solution to improve your score. If anyone manages to get the maximum score of 158, we here at *Home Computer Magazine* would like to hear from you.

All in all, this is the best game that has come across my desk in a long time. In fact, I found it difficult to tear myself away from it long enough to write this review. If you like adventure games with action, intrigue, puzzles, clues, and beautiful computer graphics, you'll love *King's Quest*. HCM

INDUSTRY WATCH

ONE YEAR AFTER THE LUBBOCK FIRE—IT'S DECISION TIME

Many consumers who purchased the TI-99/4A at a fire-sale price one year ago (when TI exited the consumer-computer business) are now making decisions regarding the fate of their entry machines. Many consoles have already been placed in new locations—collecting dust in attics and closets. Large numbers, however, are still being well-utilized with a collection of software cartridges and tape cassettes for home productivity, education, and entertainment. With the bulk of the "closeting" now past, a fairly large percentage—approaching 40% at last survey—are "adding on" in two ways: (1) purchasing a second brand of computer, and/or (2) buying up additional memory, printers, and disk drives—even spare TI-99/4A's (yes, there are some still out there) for existing systems. One result is a marked increase in disk software sales for this user base.

SOFTWARE COMPATIBILITY—THAT IS THE C-64 QUESTION

Although the new machine was announced last January, Commodore's Plus/4 computer has just recently been shipped to retail outlets around the country. The computer, priced at about $300, offers word processing, spreadsheet, graphics, and file manager programs built into its ROM. Commodore, however, is reportedly wavering on whether to make the Plus/4 software-compatible with its C-64 model—a machine which has spawned a large software base, and which will continue to spawn more software as Commodore keeps the machine alive with an expected 128K memory expansion. The off-the-shelf Plus/4 does not run C-64 software, but Commodore may be providing an add-on board if the demand for compatibility appears strong. When and how they will decide on adding the board is not known, but this "wait and see" approach to compatibility may signal a change in Commodore's tradition of making their machines incompatible with each other. Meanwhile, it appears that the Plus/4's little brother, the recently introduced Commodore 16, will have less impact in the U.S than in Europe, where it is expected to give Sinclair's Spectrum a run for its money.

PRICES ON TI CARTS TAKE A TUMBLE

In an effort to clear remaining inventory from retail channels—so as to avoid having to take back unsold inventory—TI has dropped their billed-invoice prices on most software cartridges. Some popular cartridges such as Parsec, Personal Real Estate, and Beginning Grammar, available from mail-order sources, are going for $4.95, while other titles like MicroSurgeon and Typing Tutor are being offered at under $10. As sources of TI's Extended BASIC cartridges have all but dried up, other manufacturers have stepped in. Some cartridges are licensed versions of the TI language, while others have a few differences and are not compatible with some existing TI Extended BASIC software.

TEST-DRIVEN A MAC YET? HOP ABOARD THE APPLE BUS

The Apple Bus—the network that will connect the Macintosh, IIe, and Lisa computers—is expected to be available in the first quarter of 1985. Originally announced with the Macintosh in January of 1984, development has been slow. According to sources at Apple, a laser printer with two megabytes of memory and a hard-disk file server are among the first products that will be released for this network. It is not certain, however, how long the Lisa will remain in the picture—the new Apple Bus does not require it to function. And, with a bevy of soon-to-be-released new products designed for the 512K Fat Mac—rather than Lisa—such as Lotus Development Corp.'s new integrated software package, *Jazz* ($595 including a word processor, spreadsheet, data base manager, graphics, and communications), plus an internal hard disk for a "fatter" Macintosh expected to be released sometime in 1985, we may actually see this "pioneer lady" phased out in coming months. Apple's current "Test Drive a Macintosh" campaign—in which anyone with a valid major credit card can take a Macintosh home for one night of fun and frolic—may prove to be the nails in the coffin for the more-expensive Lisa architecture.

CHIP WARS: ATARI VS COMMODORE

It's been one roadblock after another for former Commodore chairman Jack Tramiel since he took over the helm at Atari Corp. Just when he thought Atari had wrapped up a deal to buy Amiga Corp.'s graphic-chip technology, fierce rival Commodore bought Amiga, and the multi-million dollar lawsuits filed by both sides now leave it up to a judge to decide who gets what. Meanwhile, Atari has chalked up a fair measure of bad financial press as old creditors clamored for payment, and Tramiel searched for cash to develop his new products (less than $50 million of the more than $300 million in receivables Atari acquired from Warner Communications Inc. has been collected). But don't count him out yet. While he is reportedly bargaining to get more backing from Warner—and predicting to raise $150 million in the next 18 months through public and private placements—Tramiel announced that the price of the company's 800XL computers will be cut from $179 to about $120, just in time for Christmas. This may give the low-end market hotseller—the Commodore 64—a run for its money, and would also provide Tramiel with more money for investment in new products—including 8-, 16-, and 32-bit microcomputers based on a new operating system developed by Digital Research of California. Industry analysts expect these new products at rock bottom prices, as is Tramiel's style. But will firing a shotgun blast of new products scatter Atari's efforts so that nothing takes off as a best-seller? Stay tuned in 1985 as the Commodore/Atari battle rages on.

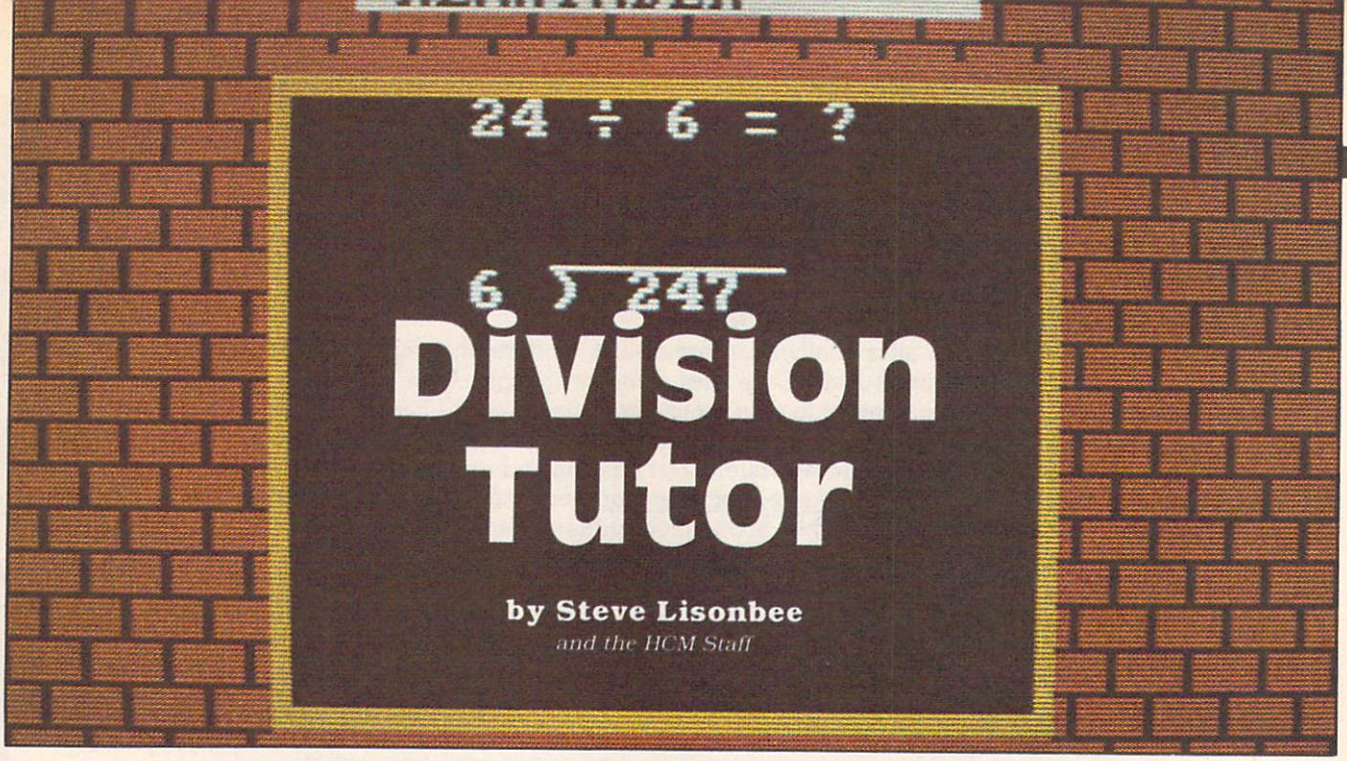
APPLE BUYERS JUGGLE PRICE VS EXPANSION OPTIONS

Many computer manufacturers would love to have Apple's problem—i.e., what to do when your product is selling too well. Apparently, Apple had expected demand for its "portable" IIc to replace demand for the IIe, and adjusted its production schedule accordingly. But instead, price-conscious consumers initially attracted by IIc ads ended up buying IIe's, and demand for the less-expensive, expandable IIe has soared. To help decrease a reported backlog of 120,000 IIe orders and ease production strains, Apple has effectively raised the price of the IIe, and lowered the price of the IIc. The price of a IIe package with disk drive and monitor was raised to $130 more than a comparable IIc system. Whether this small price difference is enough to turn the tide of IIe demand remains to be seen.

BIG BLUE OPTS FOR SATELLITES & PRICE CUTS

In an effort to retain "meaningful interaction" between students and a single instructor, a satellite-based network is helping IBM's Information Systems Group to extend its closed-circuit-television education classes across the country. IBM broadcast studios in Chicago and New York transmit signals to a satellite orbiting 22,300 miles above Earth. The data, video, and audio signals are then beamed back to classrooms at satellite-receiving centers in Dallas, Houston, Minneapolis, New York, Chicago, and soon in other cities. Students in 23 courses can communicate with instructors by using a voice-activated microphone, a call button, and a series of keys. With this kind of communication network, one has to wonder whether Big Blue's long-range goal is to eventually tie together all major computer installations in the world via satellite links—bypassing AT&T's "landline" service . . . Meanwhile, in an effort to increase "meaningful interaction" between itself and its customers and dealers, IBM has announced a wave of discounts—in the form of consumer and volume-dealer rebates—for the PC and PCjr, the PC XT, the Portable PC, and the 5152 Graphics Printer. Soaring demand for the PC AT following its introduction late this summer, coupled with the soon-to-be-forthcoming intros of the "PC2" and "lap PC," has necessitated a price-driven clearout of the older machines from existing sales channels. IBM officials are hoping that the resulting lower price point for the PCjr at the retail level will stimulate badly-needed Christmas sales, gaining market share for IBM in the home and schools. If this strategy fails, expect to see some Boca executives develop a severe case of "Charley horse."

HCM



Divide and Compute! This is the modern way to conquer the ancient practice of arithmetic.

Computers and calculators will never make good ol' arithmetic obsolete. We need to hang on to this ancient but sophisticated skill; it's something we can carry around in our heads, and with which we can accomplish many marvelous things. In fact, computers can *help* us to preserve this basic "r" (one of the "big three") by patiently exercising that biological computer—the human brain.

Past issues of *Home Computer Magazine* have included programs designed to teach addition, subtraction, and multiplication. Now we take on division—an arithmetic function that follows logically from the other three.

Division Tutor consists of two separate sections: Flashcard Division and Long Division. The first section presents simple problems with whole-number answers. The second section provides more difficult problems, to be worked out in "long-hand," which usually leave a remainder. Previous experience with subtraction and multiplication is necessary to perform these division exercises. *Division Tutor* is designed for the 8-14 age group, though certainly anyone who wants to practice their division skills may use it.

Flashcard Division

This section of the program simply displays one problem at a time and asks for a solution. Students have two chances to answer correctly. A prompt informs them of any wrong answers. After a second wrong answer, the right answer will appear, accompanied by a descending tone. If a correct answer is given, the message VERY GOOD WORK appears, along with a short musical "reward."

At the top of the screen, the numbers of right and wrong answers are displayed and updated after every problem. Near these numbers, the percentage of right answers—or "score"—is also displayed.

These flashcard exercises continue until the student chooses to return to the Main Menu.

Long Division

This section of the program takes the student step-by-step through the process of solving a problem in long division. The screen graphically depicts a schoolroom, with a problem displayed on a chalkboard. Above the chalkboard is a list of each step of the long-division process. An arrow points to the step currently being worked on. The list of steps includes:

DIVIDE
MULTIPLY
SUBTRACT
BRING DOWN
REMAINDER

". . . computers can help us to preserve this basic 'r' (one of the 'big three') by patiently exercising that biological computer—the human brain."

This part of *Division Tutor* takes the student through each step of the operation, just as the problem would be solved on paper. With helpful prompts and graphic aids, the program breaks down a complex problem into simple elements. Students enter their answers, and the program displays the numbers in the proper places. Visual aids show the student where to "bring down" numbers or place remainders.

For each step, a simple equation of the specific part of the problem being worked on is also displayed on the chalkboard, near the main problem. At the same time, the program checks for and informs the student of any wrong answers within this step. Each step allows three tries—after the third wrong answer, the program furnishes the correct answer, and then proceeds to the next step.

In the Long Division section are two difficulty levels: Beginning and Advanced. Exercises on the first level generally have single-number divisors and comprise more steps, each one relatively easy to solve. On the second level, problems more often use double-digit divisors and require fewer (but more difficult) steps to arrive at a solution. As in the Flashcard section, the

program continues to present exercises until the student chooses to return to the Main Menu.

Teaching

Of course, these and other exercises in division can also be worked out using only a stick or a piece of charcoal—this is the beauty of simple arithmetic. Stone Age man, with the right knowledge, could do as much. But computers can quickly tell you when you're right or wrong, which is what makes them an ideal *teaching* device, especially with a program like *Division Tutor*.

For *your* key-in listing see HCM PROGRAM LISTINGS Contents on page 85.

CONTROL CAPSULE
Division Tutor

KEY	FUNCTION
1	Select Flashcard Division.
2	Select Long Division.
3	Exit the program.
Esc	Return to the Main Menu.

One of the features that makes this program so much fun and easy to use is the way the Long Division option interacts with the user. The program prompts the user to enter the answer at the same location within the problem, as if the user were solving with a pencil and paper. In addition, the user can only enter enough numbers to satisfy the problem. It is impossible to enter 20 numbers to a problem with an answer only 1 or 2 digits long.

The Applesoft routine that handles this input is between lines 3070 and 3410. This routine performs limit checking on each key as it is entered. In addition, it checks for the (Esc) key (ASCII 27). If the (Esc) key is encountered, the input routine will terminate. The calling routine must then check the input for the (Esc) character so that it can return to its calling routine—generally the Main Menu.

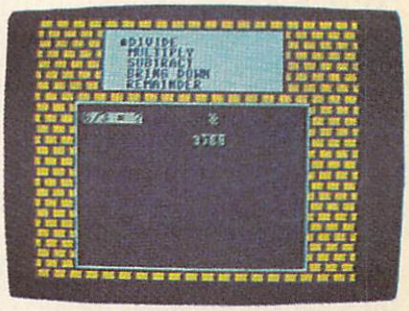
The routine also limits the user in the number of keys which can be pressed in response to an answer. Because of the strict formatting of the equation on the screen, and the user's ability to enter values right into the equation, it is necessary to limit the number of keys which can be pressed. The maximum length for any input in the routine is three (3) characters. After entering three characters, the routine will exit automatically using the characters you've entered as your answer. You can enter a value with less than three digits simply by pressing (RETURN).

CONTROL CAPSULE
Division Tutor

KEY	FUNCTION
F1	Select Flashcard Division.
F3	Select Long Division.
F5	Exit the program from the menu or return to menu from division routines.

The *Commodore Programmer's Manual* contains a comprehensive list of addresses used by the BASIC interpreter to do certain functions. One address (location 783) is used in this program to clear the STatus register before calling a machine routine. This address was mistakenly labeled in the manual as the SP (Stack Pointer) instead of the ST register.

Two other addresses are used to load the X and Y registers. These three addresses are used in this program to locate the cursor on the screen without using the PRINT statement. (See the "Home Computer Tech Note" for the Commodore 64 in this issue.) These addresses are not the actual registers in the processor—if they were, you would really mess things up by POKEing values into them at unpredictable times. Instead, when you call an assembly language program from BASIC, the system loads the values from those addresses into the the X, Y and STatus registers before entering the routine.



The beginning of a long-division problem as seen on the C-64.

CONTROL CAPSULE
Division Tutor

KEY	FUNCTION
1	Select Flashcard Division.
2	Select Long Division.
3	Exit program.
0	Return to Main Menu if entered as first response to new problem.

The IBM machines are very versatile computers, even in screen mode 0 (text mode). They have a number of useful graphics characters built into the ASCII character set. By making use of these, you gain two advantages: your program will display these special characters much faster than it could draw graphics on the graphics screens, and you have access to all sixteen colors for drawing.

A good use of this technique is demonstrated in the long-division routine. The computer generates a red brick background, complete with a black chalkboard and a white pull-down chart with the long division steps listed on it.

The brick background is created by using ASCII characters 193 and 194. These characters are placed in two strings (B$(0) and B$(1)), and then displayed on the screen:

```
FOR Z=1 TO 20:B$(0)=B$(0)+CHR$(193)+CHR$
(194):B$(1)=B$(1)+CHR$(194)+CHR$(193):NE
XT
COLOR 0,4,4:FOR Z=1 TO 12:PRINT B$(0);:PRI
NT B$(1);:NEXT
```

CONTROL CAPSULE
Division Tutor

KEY	FUNCTION
1	Select Flashcard Division.
2	Select Long Division.
3	Exit the program.
0	Return to Main Menu if 0 is the first response to a new problem.

XB *Division Tutor* requires TI Extended Basic.

The Long Division option displays a schoolroom wall complete with a red brick background, chalkboard, and pull-down chart showing the various steps in solving the equation.

Let's take a look at how easy it is on the TI-99/4A to create the background in the schoolroom. Two shapes are needed to create the staggered brick effect. We choose ASCII characters 104 (h) and 105 (i). Each character is assigned a pattern for one-half of a brick.

By staggering the characters on the screen to create a pattern, as shown below, you can create the brick pattern used for the wall:

```
hihihihihihihihihihihihihi
ihihihihihihihihihihihihihh
```

Because of the large number of graphics characters needed to define the schoolroom, it is necessary to use some of the characters in color group #8. This group includes the letters X, Y, and Z. It is necessary to use the letter Y in several of the messages displayed on the screen, which presents a problem—one character is a different color than the others. We solve this problem by substituting the letter Y for the @ symbol. This is done by getting the pattern for the letter Y with the CHARPAT statement. This statement returns the hexadecimal code used to define graphics characters. That pattern is then assigned to ASCII character 64, the @ symbol.

So, if you see text in this listing that looks like it should have had a Y in it, but you see @ instead, it's not a typo, it's there for a reason. This is the code that makes the transfer:

```
CALL CHARPAT(89,Y$)
CALL CHAR(64,Y$)
```

Division Tutor (TI-99/4A)
Explanation of the Program

Line Nos.	
100-170	Program header.
180-330	Initialize program graphics and display the title screen.
340-430	Main Menu screen.
440-460	Flashcard Division title screen.
470-510	Long Division menu screen.
520-670	Flashcard routine.
680-740	Check Flashcard answer.
750-850	Set up Long Division screen.
860-940	Select and display a problem.
950-1160	Main control loop.
1170-1250	Display the remainder. Get set up for the next problem.
1260-1310	Graphics display routines.
1320-1340	Incorrect answer routine.
1350-1430	Give the correct answer.
1440-1450	End of program message.
1460	Routine to change colors.
1470-1490	Program data.

Division Tutor (C-64)
Explanation of the Program

Line Nos.	
100-190	Program header.
200-240	Initialize program and display the title screen.
250-340	Main Menu screen. Select operation.
350-400	Initialize Long Division routine.
410-780	Flashcard Division routine.
790-1000	Display Long Division screen.
1010-2100	Main control loop for Long Division.
2110-2160	Routine to locate the cursor.
2170-2350	Input routine.
2360-2480	Incorrect answer.
2490-2590	Display the correct answer.
2600	Exit routine.

Division Tutor (Apple *II* Family)
Explanation of the Program

Line Nos.	
100-180	Program header.
190-270	Master control loop to set up and run the program.
280-440	Initialization routine.
450-570	Main Menu screen.
580-730	Menu screen graphics routines.
740-1250	Flashcard Division.
1260-1530	Long Division control loop.
1540-1730	Enter Long Division level of difficulty.
1740-1900	Display Long Division background.
1910-2000	Long Division subroutines.
2010-2690	Work the problem.
2700-2880	Display remainder.
2890-2910	Time delay loop.
2920-3060	Option to continue with Long Division or exit.
3070-3230	Input routine.
3240-3410	Read a character with a prompt.
3420-3460	Read a character.
3470-3500	Error sound.
3510-3620	Correct sound.
3630-3700	More sound effects.

Division Tutor (IBM PC, PCjr)
Explanation of the Program

Line Nos.	
100-200	Program header.
210-250	Initialize program.
260-280	Display the title screen.
290-310	Main Menu screen.
320-420	Flashcard Division routine.
430-500	Display the Long Division screen.
510-680	Main control loop for the Long Division problem.
690-720	Display the remainder.
730-740	Incorrect answer routine.
750-800	Display correct answer.
810-840	Display routines for small equation.
850-870	Display routine for the menu and borders.
880-910	Input routine for integer numbers.
920-930	Music for the correct answer.
940-970	Display the arrow in the Long Division routine.

HCM

Putting The Puzzle All Together:

Apple *IIc* Programming Considerations

By Peter Baum
and the HCM Staff

The Apple IIc provides a standard hardware environment in which to program. Understanding that environment is the key to producing effective IIc software.

In the previous issue of *Home Computer Magazine*, Peter Baum wrote an overview of the new *IIc* and promised to follow up with details on programming the newest Apple. This article is the fulfillment of that promise. Covered here are ROM and RAM memory usage, considerations in programming peripherals, interrupts, and the method for converting a DOS 3.3 formatted disk to ProDOS.—Ed.

The Memory Map

The Apple *IIc* contains 128K of RAM and 16K of ROM. Because the processor has 16 address bits, only 64K of memory can be accessed at a time. The technique used to address all of the memory is called *bank switching*. Various address ranges in memory are selected by accessing certain special locations within the memory address range. These "memory" locations perform *switch functions*—such as clearing status flags or selecting a video mode—when they are accessed in software. Because these special locations are controlled by software, they are called *soft switches*, and many are analogous to a light switch (or *hard* switch) because they have just two states: on and off. These soft switches are located in the $COOO page ($COOO-$C0FF hexadecimal) of the address space, which is reserved for these hard-wired functions instead of memory. See Figure 1 for a sample of Hardware Page address usage. A complete listing of Hardware Page address usages is contained in the *Apple IIc* Reference Manual Vol.2, Sec. B.4.

The Apple *IIc*'s two 64K memory banks are the equivalent of the main memory in the *IIe* and its auxiliary memory which resides on the Extended 80-column card. The banks are accessed using the same soft switches employed by the *IIe*. Three soft switches control reading and writing for different address ranges (see Figure 2). The RAMRD and RAMWRT soft switches control which bank of memory is accessed between the Top-of-Stack and the Hardware page. ALTSTKZP enables the appropriate memory bank for Page Zero and the Stack Page ($0000-$01FF).

Video Memories

The memory pages used for graphics ($0400-$07FF and $2000-$3FFF hexadecimal) can be overridden by another set of soft switches dedicated to video control.

In the *IIc*, 24 different display modes are possible. [See the Apple *IIc* Video Display Modes chart in "*IIc*: The Core of a New Machine" on page 13 of *Home Computer Magazine*, Vol. 4, No. 4—Ed.] These displays consist of both graphics and/or text which are generated from one of the four specific areas in memory. These areas, called display pages, are buffers where programs put data to be displayed. There are two buffers, located at addresses $0400-$07FF and $0800-$0BFF, which are used for text and low-resolution graphics. The two buffers for high-resolution graphics are located at $2000-$3FFF and $4000-$5FFF.

Graphics mode displays with text at the bottom are called *mixed mode displays*. Mixed mode displays have exactly four lines of either 40-or 80-column width at the bottom of the screen.

The text modes, both 40-column and 80-column, are supported in assembly language and Applesoft BASIC. Applesoft also contains commands that can be used to draw low-resolution or high-resolution graphics. There are no routines in the built-in firmware that support any of the double-resolution graphics modes, and the only routines that support a page 2 display are high-resolution graphics. To use the rest of the modes, such as to display the

80-column text from page 2 or to double high-resolution graphics, a software program is required. Some companies, such as DoubleStuff Software, Inc. (Doublestuff Language/Graphics Enhancer, available for $39.95, was reviewed in the Vol.4, No.3 issue of *HCM*) and ALF Products, Inc. (HGR6 Double Hi-Res Package available for $49.95) have written special programs that allow you to use these unsupported modes from BASIC.

FIGURE 1
A Portion of the Hardware Page of Memory Addresses

This table lists only the hardware locations associated with memory control in the Apple *IIc* to illustrate typical uses. The column labeled RW indicates the action to take at a particular location where:

- R means read.
- RR means read twice in succession
- N means not to read or write, because the location is reserved.

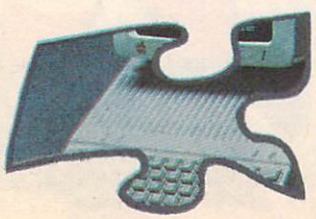
RW	Hex Address	Use
R	$C080	Read RAM; $D000 bank 2
RR	$C081	Read ROM; write RAM; $D000 bank 2
R	$C082	Read ROM; $D000 bank 2
RR	$C083	Read and write RAM; $D000 bank 2
N	$C084 through $C087	Reserved
R	$C088	Read RAM; $D000 bank 1
RR	$C089	Read ROM; write RAM; $D000 bank 1
R	$C08A	Read ROM; $D000 bank 1
RR	$C08B	Read and write RAM; $D000 bank 1

FIGURE 2
Soft Switches for RAM Access

Switch Name	Switch Address	Switch Function	RAM Address Range
RAMRD	$C002	Read main memory	$0200-$BFFF
	$C003	Read auxiliary memory	$0200-$BFFF
RAMWRT	$C004	Write main memory	$0200-$BFFF
	$C005	Write auxiliary memory	$0200-$BFFF
ALTSTKZP	$C008	Use main stack, zero page & lang. card	$0000-$01FF & $D000-$FFFF
	$C009	Use aux. stack, zero page & lang. card	$0000-$01FF & $D000-$FFFF

Firmware

For the *IIc* ROM, Apple started by modifying the firmware used in the *IIe* ROM, removing some bugs in the process, and then adding new routines to support the *IIc*'s built-in serial ports, disk drive, and mouse port. ROM in the *IIc* stretches from address $C100 to $FFFF. This ROM contains an Applesoft interpreter ($D000-$F800) and a set of routines, called the Monitor ($F800 to $FFFF), for controlling standard input/output such as the keyboard and display. The rest of the ROM firmware is used to control the system's I/O ports. Routines to support 40- or 80-column video, both serial ports, the mouse port, and the disk port occupy the ROM from addresses $C100 to $CFFF. In the *IIe* this address range is reserved for ROM on the peripheral cards. The *IIc* ROM implementation enables the software written for the *IIe* to see the built-in peripherals of the *IIc* as if they were peripheral cards plugged into the *IIe* slots.



Some of the ROM overlaps part of the *IIc*'s 64K RAM banks. The upper section of RAM memory that shares addresses with ROM is called the *language card*, and it resides in the address space from $D000-$FFFF. The language card area consists of 16K of memory, but you may have noticed that $D000-$FFFF is only a 12K address space—the language card contains two $D000-$DFFF banks, which are called bank 1 and bank 2. This memory scheme is an artifact left over from the Apple *II+*, which is also why this area is called the language card. [The Apple *II* family has evolved by building into the current model "add-on features" found in the previous model. Physically, the language card was a separate peripheral card that plugged into the *II+*, but the features contained on the card have been built into the *IIe* and *IIc*.—Ed.] The soft switches which control this part of memory allow the programmer many different memory configuration options. Figure 3 shows the soft switches associated with this memory control.

FIGURE 3
Video Soft Switch Addresses and Functions

Switch Address	Write RAM	Read RAM	Read ROM	$D000 Bank1	$D000 Bank2	Notes
$C080		X			X	
$C081	X		X			1
$C082			X			
$C083	X	X			X	1
$C088		X		X		
$C089	X		X	X		1
$C08A			X	X		
$C08B	X	X		X		1

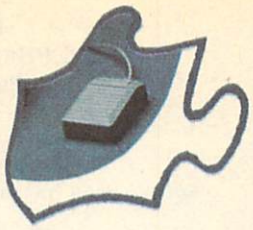
Two consecutive reads to an odd switch address write-enables RAM.

Pardon My Interrupts

Unlike the standard *IIe*, the *IIc* fully supports interrupts. On a computer, an interrupt signal is used to indicate that some device needs attention. The new firmware makes extensive use of this feature with modes that allow the mouse, serial ports, and keyboard to be run in interrupt mode. For example, when the keyboard interrupt mode is used, the processor can continue a time-consuming task without missing a keystroke. During any long operation on the standard *IIe*, the processor has to periodically check to see if a key has been pressed, or it will miss it. But on the *IIc*, the processor proceeds until a key is pressed, generating an interrupt. When this happens, the processor stops for an instant, reads and processes the keystroke, and then continues with the operation it was doing before the interrupt.

The 80-column firmware is faster on the *IIc* and contains a new terminal mode (interrupt-driven of

course) which allows the machine to keep up with 1200-baud modems.



MouseText And More

Apple has enhanced the *IIc* by adding graphics characters to the character set. These new symbols, also called icons, were designed to be used with a mouse. The icons enable programmers to offer programs with user interfaces similar to those appearing on the Macintosh and Lisa. This display mode, which is supported by the *IIc* firmware, has been dubbed "MouseText." The 32 mouse characters have ASCII codes $40-$5F when in the video memory. The icons replace a duplicate set of inverse capital letters (that are on the *IIe*) which use ASCII codes $00-$1F. If MouseText hasn't been enabled, the firmware will remap the inverse capital characters into the lower range ($00-$1F) so that the proper characters are displayed. Some Applesoft BASIC commands are shown in Figure 4 for MouseText control.

FIGURE 4
Applesoft Basic Access to Mousetext

MouseText On Mode	BASIC Command
Turn on the video firmware	[PR#3]
Enable MouseText	[PRINT CHR$(27)]
Set inverse mode	[INVERSE or PRINT CHR$(15)]
Print capital letters	[PRINT CHR$(64. . .95)]
MouseText Off Mode	
Disable MouseText	[PRINT CHR$(24)]
Set normal mode	[PRINT CHR$(14)]

In many cases, the *IIc* is compatible with the *IIe* at the firmware level. For example, the soft-switch locations for reading the mouse are completely different between the *IIc* and the *IIe* mouse card—yet Apple provided a means for programmers to write mouse-based applications that will run on both computers. They did this by specifying common entry points in the *IIc* firmware and in the firmware present on the *IIe* mouse card. The assembly language routines available through these common entry points are shown in Figure 5.

The firmware reserves specific memory locations so that it can use them to store temporary variables and results from calculations. Programs which use these reserved areas of memory do so at their own risk and may not work properly. Most of the reserved area is in the zero page (locations $0-$FF hexadecimal), but the firmware also reserves the bytes in the video buffers that aren't displayed (called *screen holes*). For example, the mouse firmware makes extensive use of these screen holes to store position and status. See Figure 6, which shows the Apple *IIc* screen holes usage of the Mouse Port. A complete list of screen hole usage can be found in the *Apple IIc Reference Manual*, Vol. 2, Sec. B.3.

"The 80-column firmware is faster on the IIc and contains a new terminal mode which allows the machine to keep up with 1200-baud modems."

Programming Graphics On The Flat Panel

The Apple flat-panel display, which will be available in the spring, has a different aspect ratio than a typical television or monitor. This is because the pixels (display elements) are square dots rather than oblong like those on a monitor. On the flat panel, the display area is three times as wide as it is high. This means that if a program draws a circle on a standard monitor, it will come out squashed when displayed on the flat panel. Though graphics programs may want to use different drawing routines with the flat-panel display, there is no way to do this automatically, since the presence of the flat panel can't be detected by a program. The program must prompt the user for this information.

FIGURE 5
Assembly Language Mouse Routines

Mouse Routine	Description
Setmouse	Initalizes the appropriate mouse mode, such as interrupt on button only, etc.
Servemouse	Service mouse interrupt, if mouse caused it.
Readmouse	Updates specific reserved locations which store current mouse X-Y position and button status. Clears interrupt status.
Clampmouse	Sets new clamping boundaries (boundaries are used to constrain the cursor within a specific window). Does not affect mouse position or update mouse position (use Readmouse).
Clearmouse	Sets the mouse position to 0, though not necessarily within clamping boundaries.
Posmouse	Sets the mouse position to new values.
Homemouse	Sets the mouse position to the upper left corner of the current clamping window.
Initmouse	Resets to startup values.

Serial Ports

While the serial ports use the same 6551 (Asynchronous Communications Interface Adapter) chip as the *IIe*'s Super Serial Card (SSC), the hardware doesn't make use of the chips in exactly the same way. Like the mouse, the only way to guarantee software compatibility between the two machines is by using firmware entry points. The commands implemented by the *IIc* serial port firmware are a subset of the SSC commands.

The firmware configures serial port 1 as a printer interface with the following default settings:

9600 baud.
8 data bits, 2 stop bits.
No parity.
80-column line width with no echo.
Line feed generated after return.
Delay after line feed of 1/4 second.
Default command character is set to CONTROL-I.

Once the printer firmware has been activated (in BASIC by a PR#1), the commands shown in Figure 7 can be used to change from the settings listed above when prefaced with the "Control I" ASCII characters. The default settings for the other serial port can be changed by prefacing the commands with the "Control A" ASCII characters as shown in Figure 7. Use this chart to help set up the serial ports for any number of devices.

FIGURE 6
Screen Hole Allocations for the Mouse Port in Main Memory

The Apple *IIc* hardware maps display-memory on the screen so that sixteen groups of eight memory locations are left over in areas of the text and low-resolution display-memory pages (for a total of 128 bytes). The *IIc* firmware uses these "holes" for various purposes. As an example of this screen hole usage, the addresses below represent functions relating to the Mouse port.

Hex	Description
$478	Mouse port: low byte of clamping minimum
$47C	Low byte of X coordinate
$47D	Reserved for mouse port
$4F8	Mouse port: low byte of clamping maximum
$4FC	Low byte of Y coordinate
$4FD	Reserved for mouse port
$578	Mouse port: high byte of clamping minimum
$57C	High byte of X coordinate
$57D	Reserved for mouse port
$5F8	Mouse port: high byte of clamping maximum
$5FC	High byte of Y coordinate
$5FD	Reserved for mouse port
$67C	Mouse port: reserved
$67D	Reserved for mouse port
$6FC	Mouse port: reserved
$6FD	Reserved for mouse port
$77C	Mouse port status byte
$77D	Reserved for mouse port
$7FC	Mouse port mode byte
$7FD	Reserved for mouse port

ProDOS: The Operating System

Because the Apple *IIc* is shipped with the ProDOS operating system, most programs developed for the *IIc* will use it instead of Apple DOS 3.3. Most BASIC and assembly language programs that were written for the *IIe* use DOS 3.3 as the operating system. ProDOS offers a significant increase in performance over DOS 3.3, including faster operation, more commands, and hard disk support. Currently, ProDOS is being included with the *IIe*, and many developers are modifying their programs to sup-

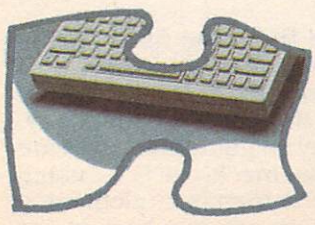


FIGURE 7
BASIC Print Strings Controlling the Two Serial Ports

PRINT "[CONTROL]IxxB"
PRINT "[CONTROL]AxxB"
Set the baud transmission rate for the port where:

xx	Baud Rate
1	50
2	75
3	110
4	135
5	150
6	300
7	600
8	1200
9	1800
10	2400
11	3600
12	4800
13	7200
14	9600
15	19200

PRINT "[CONTROL]IxxD"
PRINT "[CONTROL]AxxD"
Sets the data format of the port, where:

xx	Data Format
0	8 data, 1 stop
1	7 data, 1 stop
2	6 data, 1 stop
3	5 data, 1 stop
4	8 data, 2 stop
5	7 data, 2 stop
6	6 data, 2 stop
7	5 data, 2 stop

PRINT "[CONTROL]II"
PRINT "[CONTROL]AI"
Enables the port to echo to the monitor.
PRINT "[CONTROL]IK"
PRINT "[CONTROL]AK"
Disable the linefeed after carriage return from the port.
PRINT "[CONTROL]IL"
PRINT "[CONTROL]AL"
Enable the linefeed after carriage return from the port.
PRINT "[CONTROL]IxxN"
PRINT "[CONTROL]AxxN"
Disable the echo to the monitor and set the line length to xx.
PRINT "[CONTROL]IxP"
PRINT "[CONTROL]AxP"
Set the port's parity, where:

x	Parity
0,2,4,6	none
1	odd
3	even
5	MARK
7	SPACE

PRINT "[CONTROL]AQ"
Quit the terminal mode for port 2.
PRINT "[CONTROL]AR"
Reset port 2 and exit from firmware.
PRINT "[CONTROL]AS"
Send a 233 millisecond BREAK character.
PRINT "[CONTROL]AT"
Enter Terminal Mode with port 2.
PRINT "[CONTROL]IZ"
PRINT "[CONTROL]AZ"
Disables the [CONTROL] or [CONTROL]A.
PRINT "[CONTROL]Ixx[RETURN]"
Set printer line width.
Video echo must be disabled.

port it. It must be noted, however, that most programs on Apple DOS 3.3 formatted disks work just fine on the *IIc*.

If you have programs that are on an Apple DOS 3.3 formatted disk and you wish to convert to a ProDOS formatted disk, you need more than just the Apple *IIc*. Presently, the only ways to accomplish this task are to buy a second (external) disk drive for the *IIc* and use the utility that is provided with the ProDOS system, or to find a dual drive Apple *IIe* with ProDOS to convert the disk format—single-drive systems just will not work.

Once you have an Applesoft BASIC program converted to the ProDOS format, you will still need to comb through the BASIC code to find any command sequences that do not conform to the ProDOS structure. The most typical example is a command sequence that sets up access to a printer, and it usually takes the form of `PR# 1` under DOS 3.3. Accomplishing the same thing under ProDOS requires `PRINT CHR$(4);"PR# 1"` to be used.

If you find that a converted program doesn't work under ProDOS, you will have to locate the problem spots in the program and correct them yourself. Actually, there are very few differences to cope with, and you will find that most conversions go smoothly and quickly.

Pascal Revised

The new revision of the Pascal operating system, Pascal 1.2, will also work on the *IIc*. One of its improvements over the 1.1 version is that it utilizes all 128K of memory in the *IIc* (and in the *IIe* if the Extended 80-column card is present), enabling larger programs to be run. The new version of Pascal will also support hard disks such as Apple's 5 Megabyte Profile Hard Disk.

"If you have programs that are on an Apple DOS 3.3 formatted disk and you wish to convert to a ProDOS formatted disk, you need more than just the Apple IIc."

Applesoft Differences

Some changes were made to Applesoft in the *IIc*. Because the *IIc* doesn't have a cassette port, the Applesoft key words associated with it are no longer supported. The vectors for the key words now point to the *ampersand* vector (a location in memory containing the address of a service routine) so that you can write service routines to intercept control when any of these key words are used.

Because the Apple *IIc* has an uppercase/lowercase keyboard, Applesoft will accept lowercase input in immediate mode. When characters are typed in, the firmware will shift the characters into uppercase style. The only exceptions to this are characters typed inside quotes, `REM` or `DATA` statements, or while the program is executing (which takes it out of immediate mode).

The New Kid On The Block: 65C02

The *IIc* is based around a new extended version of the 6502 that is used in the *IIe*. This new processor, called the 65C02, uses less power (and therefore runs cooler), but is more powerful than the 6502. The 65C02 has extended the 6502's instruction set by adding several new instructions and addressing modes. These new instructions will allow programmers to write shorter and faster programs. However, if a programmer takes advantage of these new instructions, then the program will not run on a *IIe*. Because the 65C02 is just an extension of the 6502, virtually all existing software written for the *IIe* should work. The processor runs at the same speed as the 6502 in the *IIe*, performing up to 500,000 eight-bit instructions per second. [The Apple *IIc* Instruction Mnemonics chart, from our previous article, "*IIc*: The Core of a New Machine," shows the new 65C02 instructions and addressing modes available—Ed.]

The *IIc* firmware was written to take advantage of some of the new operations available with the 65C02. This allowed its programmers to squeeze as many functions as possible into the small space they were provided.

Identifying the *IIc*

If a program wants to take advantage of some of the new features in the *IIc*, such as MouseText or the 65C02, then it must have some method of identifying the machine. Apple anticipated this and has published a recommended method for identifying the different Apple *II* series computers. To identify the various machines, the program should look in the Monitor ROM locations shown in Figure 8.

FIGURE 8
Machine ID Memory Locations and their Values

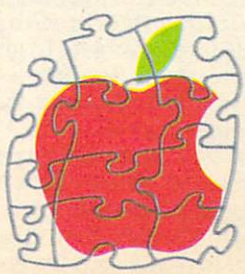
Machine	$FBB3	$FB1E	$FBC0
Apple *II* (original)	$38		
Apple *II+*	$EA	$AD	
Apple *II* emulation	$EA	$8A	
Apple *IIe*	$06		$EA
Apple *IIe* w/icons	$06		$E0
Apple *IIc*	$06		$00

Getting More Information

If all of this information has just whet your appetite and you'd like to dig deeper into the *IIc*, then I recommend buying a copy of the *Apple IIc Reference Manual* from your local Apple dealer. If you'd like to learn more about BASIC and ProDOS, then you should buy the *ProDOS Technical Reference Manual*, which is part of the Apple Workbench Series, and the *Applesoft Programmer's Kit*. Assembly language programmers should get a copy of the *ProDOS Assembler Tools* package, which is also part of the Workbench series.

Package	Part No.
1 ProDOS Tech Reference Manual	A2W0010
2 ProDOS Assembler Tools	A2W0013
3 BASIC Programming with ProDOS	A2D2037
4 Applesoft Programmer's Kit This includes package 3 above and the *Applesoft BASIC Programmer's Reference Manual.*	A2D2039

HCM



Bird Brain

You may be great at fishing, but how good are you on the wing? Learn some new angles from this flying fish-catcher—but keep your feathers dry!

by Craig Blazakis
and the HCM Staff

Watch that wave, Bird Brain! Catch that fish and fly! Here in this tropical clime, the endless surf syncopates with the rhythm of survival. Mother Frigate Bird—spying a plump, juicy prey just beneath the surface—dips down once more and times her catch to avoid the next onrushing wave. Then it's back to the nest for a quick meal and a short rest on her sun-warmed eggs before she tries again. But this time—if she makes only a slight miscalculation—her dive may end in a trip to Davy Jones' locker, as a powerful wave rears up and slaps her to the bottom. Luckily for her precious eggs, two more birds wait near the nest to take her place. And if they should meet the same fate? Well, in this case, birds, fish, and tropical beach all exist inside a computer—where reincarnation is part of the program.

Fly Fishing

At first, Bird Brain may sound easy—a real frigate bird will usually snatch a fish without even wetting a feather. But no fish is caught without effort, especially the fish in this computer game. Control the bird with joystick or keyboard. To flap her wings and give her lift, press either the fire button on the joystick or the spacebar on the keyboard. To move her left or right, move the joystick in the desired direction, or press the **S** key (for left), or the **D** key (for right). To really affect the bird's direction, press each key repeatedly, rather than pressing a key once and holding it down. This bird is subject to gravity—once she has left the nest, she must flap her wings to gain or maintain altitude. Without frequent flapping, she will plummet into the rough waters.

To make matters worse, large waves periodically travel from left to right across the screen. If the bird hits a wave, it will be knocked about, losing any control it might have had. Of course the fish, which swims along just under the surface of the water, is unaffected by the waves.

Each bird has a time limit in which it must catch a fish and return to the nest (as a real bird might exhaust its energy and fail to return). A bar across the bottom of the screen grows shorter as this time is used up. If the time limit is reached, the bird will die and take a trip to video heaven. When the bird successfully delivers a fish to the nest, the timer resets.

Of course, if the bird loses control and hits the water—through either some bird-brained maneuver, or a direct hit from a wave—it will perish.

A Heavy Load

After picking up a fish and heading for home, the bird must work harder to carry the load. If she catches the fish while descending, no amount of furious flapping will suffice to overcome her downward momentum, and she will meet a watery grave. A better method is to hover at the same level as the fish, and then start climbing as you grab it. (All that is necessary to actually "grab" the fish is to position the bird on the fish itself.)

". . . in this case, birds, fish and tropical beach all exist inside a computer—where reincarnation is part of the program."

You start the game with three birds—one in the nest and two on a branch. When all three birds have died, the game ends. But of course, you can always begin again. Scoring is based on accumulated time (in seconds) left on the clock for each fish eaten, which is then multiplied by the skill level (1, 2, or 3) for a total number of points. Your score is computed and displayed at the end of the game.

Bird Brain has three skill levels. As the skill level increases, you will find it more difficult to catch a fish and make it back to the nest. The higher skill levels have the following effects: you will have less time to catch the fish; the bird will weigh more, making it more difficult to stay in the air; and the wave will move faster.

For *your* Key-in listing see HCM PROGRAM LISTINGS Contents on page 85.

CONTROL CAPSULE
Bird Brain

KEYBOARD	
Space Bar	Flap wings
S	Move bird left
D	Move bird right

JOYSTICK	FUNCTION
Fire Button	Flap wings
Stick left	Move bird left
Stick right	Move bird right

The TI version of *Bird Brain* makes use of the best features of the TI-99/4A home computer. Sprites provide an ideal medium for the graphics animation in this program. The ability to set a sprite in motion and let it glide across the screen makes the bird in this game appear to be really flying. In addition, animation is added by changing the sprite's shape. Two sprite shapes are used for the bird while it's in flight, and a third shape for the bird at rest. The two shapes used for the bird create the animation of the bird flapping its wings.

Controlling the velocity of the bird in any direction is a simple matter. Two variables, U and S, contain the X and Y velocity. Gravity's affect on the bird is accomplished by decreasing the bird's upward motion with every pass through the program. If the variable U is decreased to the point of being a negative number, the bird will begin traveling downward.

The joystick and keyboard are checked to see if you have tried to flap the bird's wings. If you have, then L is added to U. L contains the amount of lift the bird gets for each flap. The initial value of L is determined by your playing level. The table below illustrates the lift and fall speeds for each of the 3 levels.

LEVEL	L	M	TIME
1	4	1	160
2	3	1.8	128
3	2	2	96

L = Rate of lift
M = Rate of fall
TIME = Time limit

This program proves that the Apple *II* is still a powerful graphics machine. Most programmers shy away from creating animation on the Apple using Applesoft BASIC because of its limitations in speed, and the cumbersome shape tables it requires. This version of *Bird Brain*, however, may change their minds.

Only two shapes are required to create the bird's animated flight, shapes 7 and 8 in the shape table. The current shape of the bird is contained in the variable BS. Because the bird may change its shape while flapping its wings, the variable SD keeps track of the bird's previous shape. In doing this, it's a simple matter of using the XDRAW command to erase the bird at its old location, and redraw it at its new location.

To give the bird motion, two variables containing the bird's X and Y velocity are added to BX and BY. These two variables, U for up/down, and S for sideways, are adjusted by your actions to move the bird back and forth and by flapping the wings.

The difficulty level you choose makes a major difference in this program's playing ease. Several key factors are involved, as listed below:

LEVEL	L	M	TIME
1	4	1	160
2	3	1.8	128
3	2	2	96

L = Rate of lift
M = Rate of fall
TIME = Time limit

Both IBM machines are extremely powerful graphics computers because of three versatile commands—DRAW, GET, and PUT.

The DRAW command allows you to create shapes by drawing on the screen the same as you would on a piece of paper with a pen: simply tell the computer in which direction and how far to move the pen.

Animation is made possible with the GET and PUT statements. Once you have a shape drawn on the screen, you can save it in an array with a GET statement. Here, the two shapes for the bird as well as the shapes for the wave and the fish are all stored this way. Because these shapes are not moved automatically, all of the action stops when the birds is at rest in the nest. The wave and the fish will resume their motion when the bird leaves the nest.

Once an image is stored in an array with the GET statement, you can place that image anywhere on the screen with a PUT statement. Options in the PUT statement allow you to mix the image you place on the screen with existing graphics in a number of ways. If you XOR the image to the screen, as is done in this program, the image reverses the color of any pixel on the screen that comes in contact with a pixel in the image. By writing the same image to the screen in the same location twice, you erase the first image created, and return the screen to its original state before the image was placed on the screen.

LEVEL	LIFT	FALL	TIME
1	1.5	.18	862
2	1.0	.26	602
3	0.8	.35	456

LIFT = Rate of lift
FALL = Rate of fall
TIME = Time limit

The Commodore 64 uses sprites to create the animation effects in this program. On the C-64, however, the sprites must be moved manually by the program, one pixel at a time. If you have ever tried to write a program that requires fast-moving sprites, you were probably disappointed. The program's speed limits how fast the sprites can move. This usually results in very little time being left for the program to do other functions.

For this reason we have incorporated a short machine language routine (published in *Home Computer Magazine* Vol. 4 No. 2 titled "Don't Be A Slow Poke"). This routine is driven by the system's interrupt, and automatically moves the sprites for you. All you need to do is POKE the X and Y velocity for the sprites, and the machine language routine moves them from there.

Four arrays that contain memory addresses to index the sprite motion table and the sprite location table are used in this program. The sprite motion addresses are contained in the two arrays XM( ), and YM( ). The subscripts for these arrays correspond directly with the sprite number. The values below indicate the response the sprite will have for ranges of values POKEd into the sprite motion table.

POKE VALUE	VELOCITY
0,128	No velocity.
1-127	Move to the right or down.
129-255	Move to the left or up.

LEVEL	LS	FS	TL
1	4	2	40
2	3	3	32
3	2	4	24

LS = Rate of lift
FS = Rate of fall
TL = Time limit

Bird Brain (C-64) Explanation of the Program

Line Nos.	
100-190	Program header.
200-210	Initialization.
220-390	Cleanup and status routine for sprites.
400-480	Determine the fish's color and direction.
490-640	Keep track of time.
650-710	Read keyboard or joysticks.
720-830	Check sprite position. Keep sprites on the screen.
840-1180	Control the flight of the bird.
1190-1210	Check for collisions.
1220-1230	Bird gets the fish.
1240-1250	Bird hits a wave.
1260-1360	Bird runs out of time and goes to heaven.
1370-1420	Bird crashes into the river.
1430-1450	Bird makes it back to the nest.
1460-2120	Sprite shapes.
2130-2310	Auto sprite motion routine.
2320-2460	Initialize program.
2470-2740	Title screen and skill levels.
2750-2950	Display the playing screen.
2960-3110	End of the game and score display. Option to play again.

Screen from the TI-99/4A version.

XB *Bird Brain* requires TI Extended Basic.

Bird Brain (TI-99/4) Explanation of the Program

Line Nos.	
100-170	Program header.
180-340	Display the title screen.
350-470	Input options.
480-700	Define graphics shapes.
710-980	Set up playing screen.
990-1140	Main control loop.
1150-1220	Bird splashes into river.
1230-1270	Bird lands in nest.
1280-1310	Bird hits a wave.
1320-1470	Bird gets killed. Call in the next bird.
1480-1620	End of the game. Option to play again.
1630-1760	Bird runs out of time and goes to heaven.

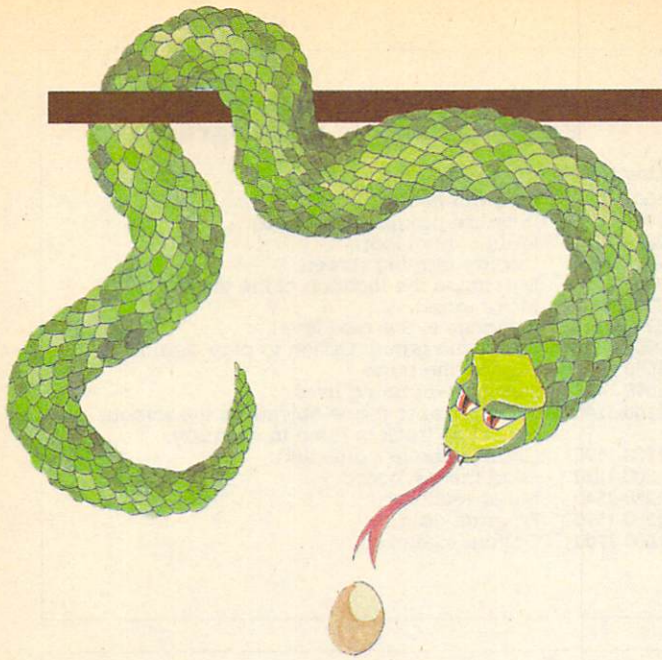
BIRD BRAIN (IBM PC, PCjr) Explanation of the Program

Line Nos.	
100-190	Program header.
200-270	Initialize program, and display title screen.
280-360	Input options.
370-460	Draw playing screen, and start the game.
470-480	Wait for the first flap to start the game.
490-520	Main control loop.
530-580	Move the bird.
590-640	Move the wave and the fish.
650-670	Read the joystick.
680-690	Read the keyboard.
700	Determine the direction of the disk.
710	Bird flaps its wings.
720-770	Bird lands in the water, new bird flies to the nest.
780-800	Bird catches the fish.
810-820	Bird hits the wave.
830-850	Bird lands in the nest.
860-880	Bird runs out of time.
890-920	End of the game. Option to play again.

BIRD BRAIN (APPLE *II* Family) Explanation of the Program

Line Nos.	
100-190	Program header.
200	Initialize program.
210-310	Subroutines for animation.
320-440	Display the title page, and input options.
450-620	Display the playing screen.
630-920	Main control loop.
930-1080	End of the game. Option to play again.
1090-1150	Bird runs out of time, goes to heaven.
1160-1310	Poke in data and shapes.

HCM



SLITHER

By Aaron Chew
and the HCM Staff

Oooooh! It's slimy! And it slithers! Yes, folks, it's a slimy, slithery, egg-eating snake on the loose—and it's growing with every bite. Just don't let it bite (or even touch) its tail, or it will literally be "the end."

Slither is a simple game, but one that has a way of "winding on." You may be surprised at just how involving and difficult it can be. On the first screen—nearly bare—the snake is quite short. Making it capture the first few eggs is an easy matter. But each time it eats one of these free-rolling eggs, the snake becomes a little longer. Finally, it becomes long enough to get in its own way—which, in this scenario, means sudden death. You must keep your snake from running into anything but the eggs themselves, including the sides of the box it is slithering around in.

Only two keys are used to control the snake's direction: G and H. The G key will turn the snake to the left, while H turns it to the right. This left/right action is from the snake's perspective, not the player's. Thus, if the snake is moving down the screen and you press the G key (for left), the snake will move to its left (your right).

Complications

Slither includes several levels of difficulty. You always start the game at level 1, with three tries (snakes). Every time you gain 500 points, you will advance to another level and add an extra snake to your "nest." Each egg is worth 20 points. If all of your snakes are wiped out, the game will end, with the option to replay.

At each higher level, the screens become more complicated. On the first level, you have only the four outside walls to contend with. Another inner wall appears in level 2. This wall is only two snake-widths from the outer wall and has only one entrance. On each new level, another wall appears—each one two snake-widths from the previous inner wall. Entrances to each inner wall are staggered top to bottom. If you were challenged by this simple game in the beginning, imagine how you will interact as it becomes more complicated.

Although *Slither* is simple in concept, it will not slide right by you. You may spend more hours mastering this game than you do some others with fancier plots and less humble beginnings.

For *your* Key-in listing see HCM PROGRAM LISTINGS Contents on page 85.

CONTROL CAPSULE
Slither

KEY	FUNCTION
G	Turn snake to its left.
H	Turn snake to its right.

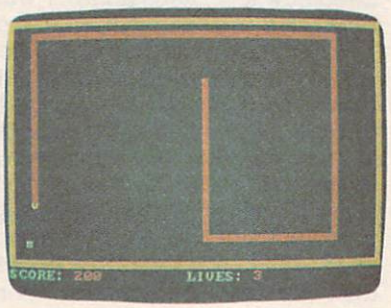
The Commodore version of *Slither* has four levels. Each new level adds another wall to the screen. Two arrays used in this program are of primary importance: The T( ) array is used to keep track of where the snake has been. Every time the snake advances, the new location is placed in the array. Every time the snake's tail moves, the cell in the array containing the location of the tail is cleared, and the variable LT (which contains the pointer to the tail) is updated. The second array is B( ). This array is a mirror image of the screen, except that the values stored there are different. This array is used to keep track of obstacles which the snake may run into. Joysticks are a bonus option for Commodore users. To turn the snake left or right, simply move the joystick to the corresponding direction.

The TI version of *Slither* has four levels of difficulty; on each level there are more walls for the snake to maneuver around. This program uses one array to track the progress of the snake. The snake is actually moved by updating the head, and erasing the tail. The index into the array for the head and tail positions are contained in the variables SP and SLA. The array SN( ) is two-dimensional, and contains the X and Y coordinates for each segment of the snake. Obstacles are easy to check for by using the GCHAR command. This command allows you to check the contents of any position on the screen.

The Apple *II* version of *Slither* includes six play levels. Each increase in level adds another wall that the snake must maneuver around. The array SN( ) is two-dimensional and keeps track of each segment of the snake. The organization of the array is 2241x2, which allows the storage of up to 2241 snake segments, each segment consisting of an X coordinate and a Y coordinate. It would be a feat indeed to fill up this entire array with snake segments. As the snake grows, more and more of the array is used to keep track of it. The snake is moved simply by advancing the head forward one screen location, and deleting the tail position.

"You may spend more hours mastering this game than you do some others with fancier plots and less humble beginnings."

There are six levels of play on the IBM version of *Slither*. Each level adds another inner wall that the snake must maneuver around. The snake is made up of segments. Each segment represents a location on the screen. These segments (screen locations) are kept in a two-dimensional integer array SNAKE%(). This array contains the X and Y coordinates for each segment of the snake. When the head is moved forward, a segment is added to the top of the array. The tail is then erased from the screen, and the last segment in the array is cleared. The color of a point on the screen is examined by using the POINT command. The color is then used to determine the type of object.



A representative screen photo from the IBM version.

Slither (IBM PC, PCjr) Explanation of the Program

Line Nos.	
100-190	Program header.
200-410	Initialize graphics characters
420-510	Display playing screen.
520-610	Initialize program variables.
620-870	Main control loop.
880-950	Completion of a level.
960-1130	Change target's position.
1140-1260	Completion of a level.
1270-1430	Title screen.
1440-1490	End of the game. Option to play again.

Slither (Apple *II* Family) Explanation of the Program

Line Nos.	
100-180	Program header.
190-250	Initialize program variables.
260-470	Main control loop.
480-730	Display playing screen.
740-760	Determine the location of the target.
770-860	Move target.
870-980	Advance to the next level.
990-1070	End of the game. Option to play again.
1080	Display the score.
1090	Display remaining lives.
1100-1140	Subroutines to place objects on the screen, and keep track of them in an array.
1150-1190	Change snake's direction.
1200-1270	Read the keyboard.
1280-1540	Music routines
1550-1590	Program data.
1600-1700	Options screen.

Slither (TI-99/4A) Explanation of the Program

Line Nos.	
100-180	Program header.
190-390	Initialize graphics and color. Display the title screen.
400-510	Initialize the start of the game.
520-610	Display the playing screen.
620-770	Main control loop.
780-990	Move the snake.
1000-1130	Snake gets the target.
1140-1410	Completion of a level.
1420-1540	Move the target.
1550-1760	Snake crashes. Option to play again.
1770-1830	Subroutine to display the score.
1840-1870	Routine to display without scrolling.
1880-2010	Subroutines for vertical printing of score, and # of snakes left.
2020-2250	Routines to display the inner walls.
2260-2290	Program data.
2300	End of the game.

Slither (C-64) Explanation of the Program

Line Nos.	
100-180	Program header.
190-250	Initialize program variables.
260-470	Main control loop.
480-730	Display playing screen.
740-760	Determine the location of the target.
770-860	Move target.
870-980	Advance to the next level.
990-1070	End of the game. Option to play again.
1080	Display the score.
1090	Display remaining lives.
1100-1140	Subroutines to place objects on the screen, and keep track of them in an array.
1150-1190	Change snake's direction.
1200-1270	Read the keyboard.
1280-1540	Music routines
1550-1590	Program data.
1600-1700	Options screen.

HCM

HOME COMPUTER™ product news

Each month we publish items of interest and news of recently or soon-to-be released computer products. Our publication of information from manufacturers of computer, peripherals, software, and accessories is not to be construed as product endorsement. Prices quoted are the manufacturers' suggested retail prices and are subject to change.

Send press releases to:

Product News Editor
Home Computer Magazine
1500 Valley River Drive., Suite 250
Eugene, OR 97401

The Great Claus Caper

Christmas Adventures On Disk

Just in time for Christmas is A Christmas Adventure, an adventure program for Commodore 64 and Apple *II* family computers. Set in and around Santa Claus' ice castle at the North Pole, the player must unravel the mystery of Santa's disappearance only hours before his annual gift-delivery run. An enhanced version of A Christmas Adventure allows users to customize the program to add personal references. The adventure alone (ACA.N) is $14.95, and the enhanced version (ACA.C) is $16.95. For $17.95 a customized version will be prepared by BitCards.

BitCards Inc.
30 West Service Rd.
Champlain, NY 12919
(514) 274-1103

❖❖❖❖

Diskette Diplomacy

Computer Diplomacy Game for the IBM PC

The board game played by Henry Kissinger and David Eisenhower is now here in computer form from the Avalon Hill Game Company. Computer Diplomacy, for the IBM PC, features a full-color screen map of Europe, which from 1 to 7 players divide up as representatives of various countries. When less than 7 people are playing, the computer fills in for the others. Using diplomatic tactics (as well as backstabbing and colluding), a player must occupy half of Europe with his or her armies and navies to win. Computer Diplomacy retails for $50, and requires 256K memory, a double-sided disk drive, and a color graphics board.

The Avalon Hill Game Company
4517 Harford Rd.
Baltimore, MD 21214
(301) 254-5300

❖❖❖❖

New TI Application Development Programs

Data Entry, Report Writing, Sketching

Easyware has announced the release of an application development package for the TI-99/4A featuring: Etch, a data dictionary program which allows users to define the parameters for their database through a menu-driven system of screens; Sketcher, a screen painter/ designer which enables users to produce quality screens with features such as menu and multi-level screens, line drawing, and color defining; Sketch, a data entry program made easy with pattern-matching selection, modification, delete and list features; and Fetch, a report writer with sorting, page headings, sort break footings, and pattern-matching options. The package is priced at $49.95.

Easyware
P.O. Box 3130 Station D
Ottawa, Ontario K1P 6H7

❖❖❖❖

Double Duty For The Commodore

Text Editor, And Graphics Kit Available

Two new software packages have been introduced by APCAD. TexED is a text editor for the Commodore 64 that functions as both a program editor and a word processor. It features both line-editing and visual-editing modes, a full-screen window, a complete range of editing functions, and a print option which allows document formatting.The TexED cassette is menu-driven and retails for $19.95. The PLOTVIC for the unexpanded VIC-20 is a high-resolution graphics kit featuring full screen window, geometric figure and text generation and positioning, eight-element color selection, three-dimensional perspective generation, and hi-res printing capacity. PLOTVIC3 and PLOTVIC8 for the expanded VIC-20 accept lightpen input and have more versatility. All three versions retail for $19.95 each.

APCAD
P.O. Box 83
Saline, MI 48176

❖❖❖❖

Tune In Tomorrow On TI

Fantasy Game Is A Continuing Saga

A fantasy adventure role-playing system in which the hero appears in a continuous saga on several game disks has been released by Creations Unlimited. Fantasy, for the TI-99/4A, features adventures in an unknown wilderness, mystical castles, magic, murky swamps, and deep caverns. Purchase of the Fantasy game system includes the first adventure, The Tomb of Gorgoroth, and costs $29.95. Subsequent story disks (for $12.95 apiece) include Where Evil Dwells, Assassin's Plot, Dragons of Doom, Curse of Targon, and Secret of Stonekeep. A disk drive and 32K Extended BASIC is required.

Creations Unlimited
20 Tilton Lane
Andover, MA 01810

❖❖❖❖

Old Stories, New Games

Children's Classics Become Text Adventures

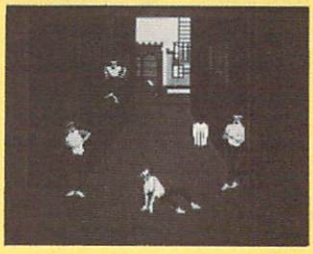
A new line of graphic and text adventures based on well-known children's classics will soon be released by Spinnaker for Apple, Commodore, and IBM computers. The Windham Classics series includes Swiss Family Robinson, The Wizard of Oz, Gulliver's Travels, Below The Root, Treasure Island, and The Wind in the Willows. The games involve characters and adventures from the above classics, and are intended to encourage players to read the books they're based on, although the books are not required to play the games. Packages retail for $26.95 each.

Spinnaker Software Corp.
1 Kendall Square
Cambridge, MA 02139
(617) 494-1200

Boot Up And Break!

A Dance-Off On The C-64

Break Street, by Creative Software, makes street gymnastics, mime, and showing-off into a computer game for the C-64. Players take on the Stingrays, a neighborhood gang, in a break dance turf battle. Using a keyboard or joystick, players manipulate the dancer through moves such as head spins, the moonwalk, snaking, and the tut. Missing a key sequence move causes a fall. Entire dance sequences may be strung together, recorded, and replayed in the future. Break Street retails for $24.95.



Creative Software
230 East Caribbean Dr.
Sunnyvale, CA 94089
(408) 745-1655

You Can Count On It

Advanced Number Skills Offered For Kids

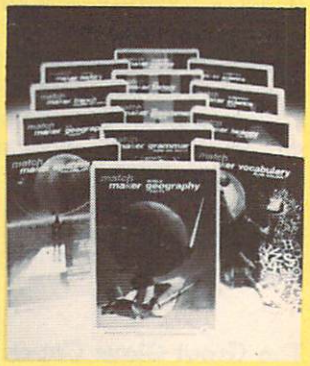
Skip Counting, a TI-99/4A program to help children build "number sense" by counting, has been introduced by B5 Software. The child selects a number to count by, the beginning and ending number of each counting sequence, and then counts by 2's, 5's, 10's, 100's, etc. Students may review odd and even numerals, multiplication products, or numeration concepts. Lessons end with a song and a graphic reward. Skip Counting is $16.95, and requires Extended BASIC.

B5 Software
1024 Bainbridge Pl.
Columbus, OH 43228
(614) 276-2752

School Subjects Hit Home

History, Biology, French Offered

The second generation of software programs in American Educational Computer's Matchmaker series (for the IBM PC and PCjr, Apple *II* family, and Commodore 64) has been released. The company claims that the new titles—U.S. Government, World History, Biology, French, and Science I, II, and III—include most subjects in standard elementary and junior high school curricula. The programs use a variety of standard quiz formats to test a student's knowledge in the various subjects. Parents may also add their own material to the programs. Visual awards and game play are offered to the student after successful completion of a question series. Each grade level program retails for $39.95.



American Educational Computer, Inc.
2450 Embarcadero Way
Palo Alto, CA 94303
(415) 494-2021

Big Blue Belts Out A Few

IBM Rolls Out 10 New Home Programs

IBM has introduced 7 new entertainment programs and 3 educational programs for their PC and PCjr computers. The entertainment programs include Trivia 101: The Introductory Course, and TV and Cinema 101, each with more than 5,000 questions; PC Pool Challenges, a simulation of straight pool and eight ball billiard games; Touchdown Football, a football simulation program that is only available for the PCjr; Jumpman and Shamus, two arcade games; and Zyll, a text adventure game set in the medieval ages. Jumpman, Shamus, and Zyll have a suggested retail price of $35, and the others, $30.

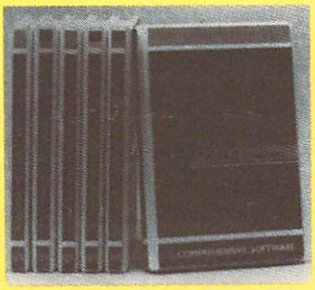
The educational programs are part of IBM's Electric Literature Series, based the new Electric Poet program. Electric Poet ($75) gives teachers and parents the ability to create animated, musical lessons for language arts, social studies, math, science, or other topics. Comma Cat and Dictionary Dog, $45 each, were designed using Electric Poet and use its interactive teaching capabilities to teach the use of punctuation marks and the dictionary, respectively.

IBM Entry Systems Division
P.O. Box 1328
Boca Raton, FL 33432
(305) 982-3474

Have We Been Introduced?

Programs Teach Common Applications

The Intro Series, a set of user-friendly programs designed for the IBM PC and PCjr to teach users about the various functions of personal computers and common applications programs, has been introduced by Comprehensive Software. Included in The Intro Series are Introduction to Personal Computing, Introduction to the Operating System, Introduction to Databases, Introduction to Communication, Introduction to Accounting, and Introduction to Electronic Spreadsheets. Each program retails for $59.95



and comes complete with examples of applications areas from popular programs such as Lotus 1-2-3 and dBASE *II*.

Comprehensive Software
2810 Artesia Blvd.
Redondo Beach, CA 90278
(213) 214-1461

Just a Phone Call Away

Convert the C-64 Into a Smart Terminal

Phone Call, a telecommunications program for the Commodore 64, has been introduced by Arrays, Inc./Continental Software. The program converts the C-64 into a "smart" terminal capable of performing a variety of transactional operations such as home banking, electronic mail, data retrieval, travel planning, and more. It features help screens, and permits uploading and downloading of machine language programs. Phone Call is available for $49.95.

Arrays, Inc.Continental Software
11223 South Hindry Ave.
Los Angeles, CA 90045
(213) 410-3977

Star-Struck Computing

Database System For Astronomers

For astronomy buffs, Astrobase by Zephyr Services offers a mini database system that comes with the 300 most important astronomical objects of the sky beyond our solar system. Objects include galaxies; open and globular star clusters; dark, emission, and planetary nebula; quasars; etc. Users can add additional objects, do searches for deep sky objects, print results, and add observation notes. Astrobase is available for $29.95, and runs on IBM PC and PCjr and Apple *II* systems.

Zephyr Services
306 South Homewood Ave.
Pittsburgh, PA 15208
(412) 247-5915

Apple's New Blossoms

RGB Monitor, Enhanced AWII Released

Apple Computer has released some new products for the Apple *IIe* and *IIc*, including a 12-inch RGB monitor. The AppleColor Monitor 100 for the *IIe*, *III*, and *III* + features a screen-tilting mechanism that allows users to adjust the viewing angle, and an anti-reflective screen surface that reduces glare. It also has a monochrome switch. Apple's new Extended 80-Column Text/AppleColor Card is required to connect the monitor to the computer. The card adds 64K of internal memory in addition to providing color output. An RGB color adapter that will allow the monitor to be used with the Apple *IIc* will be introduced by Apple later this year. The monitor retails for $599, and the AppleColor Card retails for $299.

An enhanced version of Apple Writer *II* for the *IIe* and *IIc* has also been released by Apple Computer. New features of the ProDOS-based Apple Writer *II*, Version 2.0 include horizontal scrolling, text display that lets users see the page and line count without printing, a built-in terminal mode that allows users to access information services, and a utility that enables users who do not have a ProDOS user's disk to format a blank disk for use with the new Apple Writer *II*. It can work with Apple's ProFile hard disk and Quark Inc.'s Catalyst *IIe*, a program selector that allows all *IIe* programs to be stored on a hard disk. Apple Writer *II*, Version 2.0 sells for $149, and upgrades for those with the previous version of the program cost $50.

Apple Computer, Inc.
20525 Mariani Ave.
Cupertino, CA 95014
(408) 996-1010

Color Your Apple Inexpensive

Integrated Apple Color Graphics Package

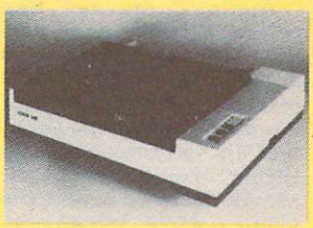
Flying Colors with Printout Program, an integrated color graphics, slide projector, and printout software package for the Apple *II* +, *IIe*, and *IIc*, has been released by The Computer Colorworks. The program allows users to create color graphics with an interactive drawing program and then to print out their creations onto any of 37 dot-matrix printers. Included in the $69.95 package is a paint program (free-hand drawing, automatic line, box, and circle routines, and text) and a

slide projector program which enables the user to store, recall, organize, and display pictures.

The Computer Colorworks
3030 Bridgeway Suite 201
Sausalito, CA 94965
(415) 331-3022

A Legend In Printing

New Printer Has Square Dots



Cal-Abco has announced the Legend 880 dot-matrix printer, which uses a new square dot to produce high print quality. The printer prints 80-column lines at 80 characters per second, and offers over 40 different character fonts, mixable on a single line. It is bi-directional and comes with an 8-bit standard Centronics Parallel interface. In addition, the Legend 880 generates 228 ASCII characters and high-resolution graphics with a 9-wire print head warranted for 50 million characters. The printer retails for $279.

Cal-Abco
14722 Oxnard St.
Van Nuys, CA 91401
(818) 994-0909

Precise Calculations

A Professional Equation-Solver

A program that interactively solves science, engineering, and business equations is available from Interactive Microware, Inc. Varicalc, for the Apple *II* family of computers, will solve for any one of 19 variables on the right or left side of a formula without rearranging the formula. Results can be plotted on the screen or printed, and up to 255 equations may be stored on disk for recall. Varicalc is priced at $100.

Interactive Microware, Inc.
P.O. Box 139
State College, PA 16804-0139
(814) 238-8294

Guarding Against Modem Hangups

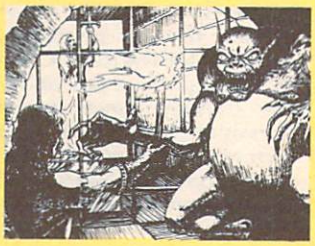
Set Up An Automatic Dedicated Line

Dataguard will prevent accidental data loss or disconnection of modem users by setting up an automatic dedicated line. Control Industries' product gives the modem user priority on the line when another phone is picked up, without disruption of normal telephone functions. Dataguard comes as an In-Phone model, or as a 12-foot Snap-In-Cord model which replaces your present phone cord. It is FCC approved and retails for $39.95.

Control Industries
Box 6292
Bend, OR 97708
(503) 389-1969

Watch Out For That . . .

Hole in the Ground Is First Of Adventures



Tesseract Software has announced the first in their series of interactive graphic adventures for the TI-99/4A. The first program, Hole in the Ground, is programmed in TI BASIC and includes a recorded oral explanation and set-up of the game. It is priced at $15.

Tesseract Software
701 Park Lane
Derby, KS 67037

Pop Goes The IBM

Desk Tools Available For The PC/PCjr

Bellsoft has introduced Pop-Up desk tools, a series of programs that, once loaded, can be accessed with one keystroke while the user is running other applications. The first six Pop-Ups, which run on the IBM PC and PCjr, are Calculator, Notepad, TeleComm, Alarm Clock, Calendar, and PopDOS.

Pop-Up Calendar, like the other programs, pops up in its own window at any time, even when users are running other applications. It displays 3 months at a time for any year, or one month with holidays and appointments marked. It retails for $19.95.

PopDOS allows users to access DOS commands at any time. Users can look at a directory, copy, rename, delete, or print files in a choice of typefaces, as well as check on how much memory is available. It costs $39.95.

Pop-Up Notepad, also $39.95, is a scratchpad program that lets users jot down quick notes, create lists, and keep track of things. It moves information between applications, and has the same editing features of a word processor. Notepad notes can be stored as permanent files.

Pop-Up Calculator, $39.95, works like a real calculator, operates on values in up to 10 memories, and displays results that can be printed. It will also pass the results of its calculations to the user's other applications.

With Pop-Up TeleComm, users can dial their most-often-called phone numbers automatically, enter them at the keyboard, or take them from a phone list or database. Users can instantly connect to their choice of information services, bulletin boards or other computers, and receive and transmit information without leaving the program they're running. TeleComm is $79.95.

Pop-Up Alarm Clock displays the time, sets alarms with reminder messages, and lets users run programs when they're away from the computer. It has a stopwatch, and copies are being offered free to IBM users as a promotion.

Bellsoft, Inc.
2820 Northup Way
Bellevue, WA 98004
(206) 828-7282

Running On Adventure

Escape The Kryon Empire

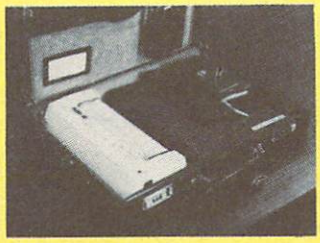
EB Software has announced TI Runner, a program written in assembly language with music, a demo mode, and graphics/animation. As the TI Runner, the player is a highly trained commando who has been captured and imprisoned in the Kryon Empire. The TI Runner must conquer 50 levels while avoiding guards and collecting treasure on the way to the surface and freedom. The game retails for $24.95 and requires an Editor/Assembler or Mini-Memory module, and 32K memory expansion.

EB Software
12912 Villa Rose Dr.
Santa Ana, CA 92705

Shh! It's The HUSH 80

A Lightweight, Low Cost Printer

An 80-column, 28-ounce, dot-matrix thermal printer for $159.99 has been announced by Ergo Systems, Inc. The HUSH 80 portable thermal printer features 80-column, bidirectional printing at 80 characters per second, and graphics at 4800 dots per square inch. It comes in three models, each of which can be equipped with a built-in rechargeable nickel-cadmium battery pack. The HUSH 80CD provides direct interfacing to the Commodore line of computers, the HUSH 80P has a Centronics-type parallel



interface, and the HUSH 80S provides a serial RS232 interface. All models include the interface, interface cable, 100-foot roll of thermal paper, and a 9-volt a.c. wall transformer with power cable, and will fit into a conventional briefcase.

Ergo Systems, Inc.
1360 Willow Rd.
Menlo Park, CA 94025
(415) 322-ERGO

Keep Your Dungeons To Yourself

An Integrated Game-Creation Package

New for TI-99/4A gameplayers is Dunjon Creator, a package of three integrated programs: Dunjon Definition, a character editor which gives the user complete control over the color and definition of monsters and other features; Dunjon Maker, which allows a floorplan to be designed using the defined characters; and Dunjon Play, which may then be used to form a user-defined world and move a hero through it. The game requires Extended BASIC, and retails for $17.95 for the cassette version, and $19.95 for the disk version.

Phoenix Computer Enterprises
8 Jay Circle
Windsor, CT 06095

At The Tone, Daylight Time Will Be . . .

Card Adds Clock, Calendar, Print Spooler

Legacy Technologies, Ltd. has introduced The Legacy CPS multi-function expansion card for the IBM PCjr L-Bus. The card adds a clock, calendar, and parallel printer port to the PCjr through the Legacy expansion system. When the system is off, a battery powers the clock/calendar. Software that comes with the card allows the user to automatically set time and date, set system prompt with time,

and print spooler. The print spooler allows the CPU to be "slaved" for printing functions so that the user can perform 2 functions concurrently on the Junior.

Legacy Technologies, Ltd.
4817 North 56th St.
Lincoln, NE 68504
(800) 228-7257

A Solar Shower Of "Edware"

Sunburst Adds 3 To Educational Line

Sunburst Communications, Inc. has added three programs to its home educational software line. The Incredible Laboratory, for ages 7 to adult, uses trial and error and note-taking skills to discover what combinations of mysterious chemicals make up crazy monsters. Challenge Math, for ages 6 to 11, contains three programs that let children practice basic math, estimation, and problem-solving skills. Getting Ready to Read and Add gives preschoolers a chance to practice letter and number recognition. Each program is $39.95, and they are all available for Apple *II* systems. Challenge Math is also available on the Commodore 64.

Sunburst Communications Inc.
Pleasantville, NY 10570
(800) 431-6616

Achtung! Some WW II For Big Blues

Castle Wolfenstein Now Out For PC/PCjr

Following three years of popularity on other systems, Castle Wolfenstein by Muse is now available for the IBM PC and PCjr. The player assumes the role of a World War II G.I. who has been captured behind enemy lines and taken to an ancient castle to face interrogation and death. The G.I. must find the Nazi war plans, battle guards, and escape from the fortress. Castle Wolfenstein costs $29.95 retail.

Muse Software
347 North Charles St.
Baltimore, MD 21201
(301) 659-7212

Setting the ProDOS Prefix

Apple's Professional Disk Operating System (ProDOS) for Apple II Family computers introduces a new complexity to Apple disk file management. When accessing a particular disk file under previous Apple Disk Operating Systems, a disk need only be identified by the slot and drive number that contained the disk—it didn't matter which disk was in the drive because a disk's identity was not relevant to the system. But under ProDOS, each disk has a distinctive volume name, and disk files are identified by their "Pathname," which begins with the volume name.

The volume name (or any directory or file name) in ProDOS is delimited by a slash character at its beginning and end. The name must begin with a letter and may contain letters, numbers and periods—no spaces or other punctuation are allowed. The volume name is the first part of what is called a prefix, and ProDOS always keeps track of the current default prefix that it uses in disk access. If you wish to access a disk with a name different from the default prefix, it must be specified by name. So how can programmers get their programs to access a disk whose name is unknown? You can still access a disk by slot and drive number to maintain compatibility between DOS 3.3 and ProDOS, which is the key to answering the question.

If you execute a **PREFIX** command from a program without specifying a slot or drive, the next **INPUT** statement places the current ProDOS prefix in whatever variable is specified. For example, if the current prefix is /MY.DISK/ and the following two lines are executed:

```
10 PRINT CHR$(4);"PREFIX"
20 INPUT PF$
```

then **PF$** = /MY.DISK/. If, however, line 20 were deleted and line 10 were changed to read:

```
10 PRINT CHR$(4);"PREFIX,D1"
```

then the default prefix is changed to the volume name of the disk in drive 1.

For an application such as the Quiz Construction Set programs (on pages 14-22 of this issue of HCM) where it is desirable to store the same file on two separate disks (such as putting a copy of a quiz on both the disk that contains Quiz-Make and the one containing Quiz-Take), it is essential that the prefix be altered from the program.

In DOS 3.3, the programmer does not have to worry about which disk is in the drive. When a file is to be saved or read, the user just places the disk containing the file into the drive and the program will access it no matter which disk is in the drive. A simple error-handling routine is necessary to see whether the file already exists on the disk. In ProDOS, however, an extra step is added—the prefix of the data disk must be determined before the file can be accessed properly.

This is easily accomplished in the ProDOS enhancement of Quiz-Make by instructing the user to place the disk in drive 1 and execute a **PREFIX** command designating a particular drive:

```
836 PRINT : PRINT "INSERT DISK IN DRIVE 1 AND PRESS A KEY" :GOSUB 1750: D$=CHR$(4)
837 PRINT D$;"PREFIX,D1"
```

The **GOSUB 1750** in line 836 of the program branches to a keyboard input routine that waits until a key is pressed before the program continues. The program effectively halts until the user signals that the proper disk is in drive 1. The **PREFIX,D1** command changes the default prefix to the volume name, no matter which disk has been placed in drive 1. Any time the programmer expects the user to change disks, this line should be included to reset the default prefix, and assure that a PATH NOT FOUND error does not occur when other disk commands—such as **VERIFY**, **OPEN**, **READ**, **WRITE**, **CLOSE**, etc.—are executed.

This procedure may appear to be a bit cumbersome to someone accustomed to DOS 3.3. However, with a few tricks like this one, ProDOS is actually a more bug-free and controllable programming environment.

—Roger Wood

Cursor Movement Made Easy

Many BASIC languages contain specific statements that will place the cursor at a particular line and column on the screen before **PRINT**ing a message or accepting **INPUT**. Instead, C-64 BASIC uses the **PRINT**ing of the cursor key functions in a quoted string to handle this type of function. Although this is generally a handy way to control cursor movement, a simple "Move the cursor to row 4, column 5," would certainly make writing some BASIC applications a lot easier.

The most-often used way to move the cursor to a specific spot on the screen is a little cumbersome, but reliable. Just enter a **PRINT** statement in your program, followed by the double-quote mark. Then press the **[HOME]** key, and a reverse-color S will appear. If you finish the statement with another double-quote, the statement will cause the cursor to go to the top row, far-left column when the statement is executed, without altering the display. To move the cursor to row 3, column 17, you would follow the **[HOME]** key-press in the **PRINT** statement described above with two cursor-downs and 16 cursor-rights, and then close the quotes. Although this method works, it does make for some rather long **PRINT** statements in a program.

A much easier way is to **SYS** (or call) a machine language kernal routine called PLOT. This routine is located in kernal ROM at address 65520 decimal ($FFF0) as described in the Commodore 64 Programmer's Reference Guide on page 290. The description says that to move the cursor to a specific location, you must clear the carry flag, set the .X and .Y registers to the row and column of the desired cursor location, and call the routine. This might at first seem difficult—but thanks to 4 storage locations in memory, it's really quite easy. According to the memory map on pages 310-334 of the guide, the 4 memory locations used by the kernal to provide access to these registers from BASIC are:

Decimal Address	Hex Equivalent	Storage For
780	$30C	.A register
781	$30D	.X register
782	$30E	.Y register
783	$30F	.SP register

These locations are not the actual registers themselves; whenever a command is executed from BASIC, the kernal loads the values from these memory locations into the registers before it actually links to the machine language routine.

Because of the abbreviation used, the .SP register above would appear to be the Stack Pointer. But, if you changed the Stack Pointer every time you executed a **SYS** command, the results would be catastrophic because the 6510 processor in the C-64 uses the stack to keep track of where it comes from each time it executes a **SYS**. The .SP register is actually the processor's status register, which is just the register you need to access in order to clear the carry flag before calling the PLOT routine.

The BASIC code needed to use the PLOT routine is a total of 4 statements: 3 **POKE**s and 1 **SYS**. Here's a line of code that would move the cursor to row **R**, column **C**, where **R** is a value between 0 and 24, and **C** is between 0 and 39:

```
POKE 781,R:POKE 782,C:POKE 783,0:SYS 65520
```

This line can be used as a subroutine in any BASIC program. Before executing a **GOSUB** to it, the values of **R** and **C** are set to the Row and Column where the cursor is to be moved. Note that the top row of the screen is row 0, and the bottom row is row 24. Likewise, the left column is column 0, and the right column is column 39. To see exactly how this routine can be used, look at the C-64 Division Tutor program (lines 2140-2160) in this issue of HCM.

One word of warning: if you place a value outside of these ranges in the variable, you'll get unpredictible results. Any number greater than 255 will cause the program to "bomb," returning an illegal-quantity error.

—Roger Wood

Redirecting I/O From DOS

If you are using DOS 2.0 or 2.10, you can redirect the input or output of your system to another device without affecting your application. This is possible because almost all applications use DOS function calls to handle input and output. The standard input device (abbreviated **STDIN**) is the keyboard, and the standard output device (**STDOUT**) is the screen. Redirect I/O by telling the computer to which device **STDOUT** and **STDIN** will be connected.

To redirect **STDIN** and **STDOUT**, you must be at the DOS level of your computer—no applications may be running, including BASIC or BASICA. If you are unsure about how to get to the DOS system, call up BASIC, and then enter **SYSTEM**. (You will know you're there when the **d>** prompt appears. Here **d** is the disk drive the system is currently using as the default drive.)

Now enter a redirection command. If you have a printer, try entering **DIR>PRN**. The directory of the disk currently in the default drive will print to your system's printer. The < (less than) and > (greater than) symbols are used to redirect the input and output. In the above example, the output from the **DIR** command, which is a list of the files in the directory, is redirected to **PRN**, which is the printer. Output of the **DIR** command could just as easily be redirected to a disk file by substituting the file name for **PRN**. If you redirect to a file, the file will be reset to a length of zero, with the output of the redirection going to the first record in the file. To append data to an existing file, use a double greater-than sign (>>) instead.

Programs which normally output to the screen can be redirected to the printer, or any other device. This technique might be used to get a hard copy of the line numbers executed when a program runs. Redirect the output with a statement like this: **BASIC>PRN** (or **BASICA>PRN** if you're using BASICA on the PC). Load the program and enter the command **TRON**, then run the program. The line numbers normally printed to the screen during the trace will now appear on the printer. Anything that normally would have been printed to the screen will echo to the printer as well. When you want to redirect the output back to the screen, return to DOS with the **SYSTEM** command and then re-enter BASIC without the redirection.

You can also redirect the input device to your application. You can directly access BASIC, redirecting input from a file to the BASIC interpreter. By redirecting input to the BASIC interpreter directly, you can create batch jobs controlled by the file—you might want the file to run several programs consecutively, or you may have several different jobs for one program requiring several different sets of inputs. Set up a separate file for each case, or chain together the programs from one file by having the file **LOAD** and **RUN** each program. If no program is running under the BASIC interpreter, the interpreter itself gets its commands directly from the file (these could be any legal commands typed in from the keyboard). If the file tells BASIC to **LOAD** and **RUN** a program, it will get its input from the file as soon as the program begins executing.

If the program halts, then the BASIC interpreter goes back to the file again for more direct commands. This will continue until the interpreter encounters a **SYSTEM** statement, or the end of the file is reached. If the end of the file is reached, you will get the error message INPUT PAST END, and the system will exit to DOS. You must always return to DOS before you can redirect the input back to the keyboard. The redirection will last only as long as the current application (such as the BASIC interpreter) is running.

—William K. Balthrop

Doing Without Extended BASIC "ACCEPT AT"

A very useful feature missing from TI BASIC is the ability to accept input from any location on the screen. The **INPUT** statement only allows information to be typed at the bottom of the screen. The TI Tech Note of the Vol. 4, No. 2 issue of HCM showed, among other things, how to display information on the screen at any location. Here is a routine that will input information from any line on the screen.

The only drawback of this routine is its absence of sophisticated editing keys, which are available when using the **INPUT** statement. However, the left and right cursor control keys move the cursor left and right anywhere on the line, so you can type over mistakes. This subroutine is a one-line input routine, meaning that you can't type in more than one line at a time without recalling the routine again.

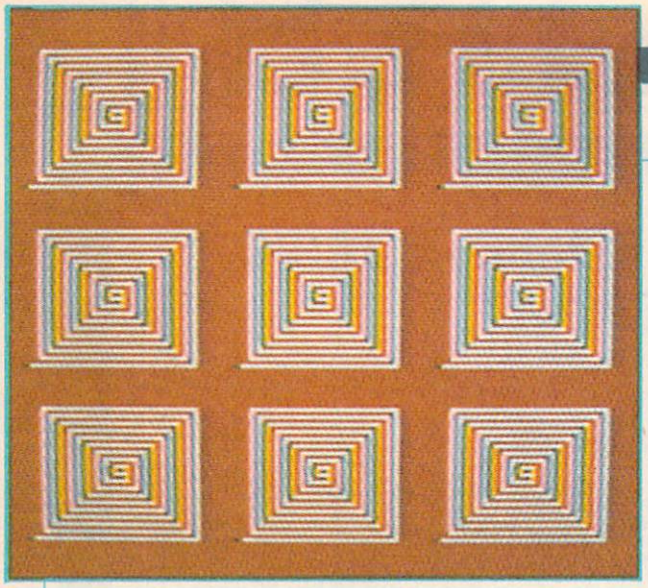
To use this routine, set up the value of **XP** to indicate the input line. Branching to the subroutine without assigning a value to **XP** will produce an error message. To terminate input, simply press **[ENTER]**—the input will be returned in **K$**. To illustrate how this routine can be used we have supplied a short demo in the beginning of the program. The demo will clear the screen, set the row pointer **XP** to 12, and branch to the Input routine. After coming back from the routine, the line entered will be printed, and the program will halt.

```
  100 REM  * DEMO FOR INPUT ROUTINE *
  110 CALL CLEAR
  120 XP=12
  130 GOSUB 10000
  140 PRINT K$
  150 END
  160 REM
10000 REM  * INPUT ROUTINE *
10010 CP=1
10020 K$=""
10030 CALL GCHAR(XP,CP+2,C)
10040 CALL KEY(0,K,S)
10050 CR=ABS(CR-1)
10060 IF CR=1 THEN 10090
10070 CALL HCHAR(XP,CP+2,95)
10080 GOTO 10100
10090 CALL HCHAR(XP,CP+2,C)
10100 IF S=0 THEN 10040
10110 CALL HCHAR(XP,CP+2,C)
10120 IF K=13 THEN 10340
10130 IF (K<>8)+(CP=1)THEN 10190
10140 CP=CP-1
10150 CALL GCHAR(XP,CP+2,CH)
10160 K=CH
10170 GOSUB 10350
10180 GOTO 10030
10190 IF (K<>9)+(CP=28)THEN 10260
10200 CP=CP+1
10210 CALL GCHAR(XP,CP+2,CH)
10220 K=CH
10230 IF CP<LEN(K$)+2 THEN 10260
10240 K$=K$&CHR$(32)
10250 GOTO 10030
10260 IF CP-1=LEN(K$)THEN 10310
10270 GOSUB 10350
10280 IF CP=28 THEN 10030
10290 CP=CP+1
10300 GOTO 10030
10310 K$=K$&CHR$(K)
10320 CALL HCHAR(XP,CP+2,K)
10330 GOTO 10280
10340 RETURN
10350 K$=SEG$(K$,1,CP-1)&CHR$(K)&SEG$(K$,
      CP+1,LEN(K$)-CP)
10360 CALL HCHAR(XP,CP+2,K)
10370 RETURN
```

***Input Routine* (TI BASIC)**
Explanation of the Program

Line No.	
100-160	Demo routine.
10000-10020	Initialize the routine.
10030-10100	Flash cursor and wait for a key-press.
10110-10120	Check for the return.
10130-10180	Left cursor routine.
10190-10260	Right cursor routine.
10270-10300	Housekeeping on the line.
10310-10340	Add a character to the end of the line (append).
10350-10370	Insert a character into the line (type over).

—William K. Balthrop



LOGO CLONES:

TI Graphics in a Turtle-Shell

by Sidney D. Nolte

Learn the secret of the turtle clones! Duplicate turtles draw on the screen —with a single command.

Have you ever wondered why the message OUT OF INK occurs in TI LOGO? It happens because only 256 characters can be used at any one time in graphics mode. The standard character set (tiles 32 to 96) removes 64 of these, leaving only 192 tiles for displaying all other graphics. When all 192 tiles are used, the OUT OF INK message appears. By employing the following procedures, you can use this apparent disadvantage to display some spectacular and unusual turtle graphics.

The CLEARSCREEN command clears all graphics and characters off the screen, except sprites. It actually fills the entire screen with the SPACE character (tile 32). When the turtle draws on the screen and there is a SPACE tile where it is instructed to draw, it replaces the SPACE with one of the 192 tiles available for graphics. At this time the new tile is cleared of any pattern that it might have contained, so that only what the turtle draws appears in that tile. When the turtle draws on the screen and encounters a tile other than a SPACE tile, it draws on that tile without erasing it first.

Now that we know about this unique scheme for drawing, we can place the same tile in several locations on the screen, and by drawing on that tile with the turtle, we cause the turtle's drawing to appear in each location that contains that tile. Try this short experiment on your system:

```
TELL TURTLE
CLEARSCREEN
PUTTILE 0 16 11
PUTTILE 0 10 10
FORWARD  10
```

Surprise! The FORWARD 10 command drew two lines because tile 0 was on the screen in two different locations. If the screen had been filled with tile 0's, then you would have filled the screen with short lines.

The procedures listed below use this technique to create two different effects. The first procedure, STARS, draws a five-pointed star within a square area. The number of tiles on one side of the square is passed to it as the parameter M. With these procedures in memory enter:

```
STARS 4
```

There will be a little wait while the screen is prepared, but be patient—it's worth the wait. The result is many stars—each one within its own 16-tile area—being repeatedly drawn and erased throughout the screen.

The second procedure, called MANY, accepts two parameters: the length and the width of a rectangular area in which patterns will be drawn. For example, a 6 by 3 tile area would be used when you enter:

```
MANY 6  3
```

The key to success of these procedures is recursion. The DRAW procedure repeatedly calls itself, drawing and then erasing spirals on the screen. This effect is most spectacular when the spiral excedes the size of the character blocks on the screen. The spiral actually spills over into an area which is a copy of what was already drawn, and the resulting patterns can be hypnotizing.

HCM

Build a LOGO Adventure

by Andrew Keith

and the HCM Staff

Learn how to create your own interactive fantasy world in LOGO. Part 1 zeros in on getting your computer to understand and respond to English.

Without a doubt, one of the most remarkable things that a computer can do is to present us with wonderful fantasy worlds to explore. The setting may be in a foreboding old castle with vampires and witches, or on an uncharted planet in deep space. The goal may be to escape from an evil archfiend, explore the ruins of an ancient civilization, or amass great wealth by recovering rare and priceless treasures. We're talking, of course, about the genre of computer games called "interactive fiction," or adventure games. Adventure game programs create the illusion that the computer understands English. You can type GIVE THE SWORD TO THE WIZARD, and the program will respond by attempting to carry out your wish. Depending on how the programmer wrote the story, your command will either advance the plot, get the main character killed, or have no effect at all. Taking part in a computer adventure is like reading a book in which *you* are the main character, and *you* make decisions which can drastically affect the outcome of the story.

The computer is seemingly able to understand English because all adventure game programs include something called a "parser." My dictionary defines parse as: "to give a grammatical description of a word or group of words." That is exactly how the procedure TO PARSER in our LOGO adventure is able to make sense of what you type in. In the example above, it would look at the positions of the words in the sentence and make a determination that GIVE is a verb and SWORD is a noun, the thing being given.

LOGO: Perfect for Adventure

One forte of LOGO is its handling of words and sentences, so it's the ideal language for writing an adventure game. We will begin by writing a procedure called GETCOMMAND that will take any command that is typed in and hand it to the parser to be analyzed. In later installments, we will look at how to feed the computer the data it needs to set up an adventure world of your own design, and at how to use list processing techniques to manipulate objects within that world. By the time we are done, we will have constructed an entire interactive fantasy in LOGO, and will have laid the groundwork for building countless other scenarios.

The pattern for the GETCOMMAND procedure will be as follows: get a command, get the parser to break it down into a noun and a verb, and then RUN the verb as a procedure (with the noun, if there is one, as an input to that procedure). Finally, GETCOMMAND will loop back to the top of the procedure for another go around. The advantage of this construction is that it doesn't matter how long or involved our sentence is, as long as the parser can pick out the two elements it needs. We can type out lengthy sentences, or simple verb-noun commands like TAKE CROWN, and our parser will respond appropriately either way. Let's examine how it works.

Notice that the body of the GETCOMMAND procedure is enclosed in a loop using the GO primitive. The Apple LOGO *II* and IBM versions declare a label with the reserved word LABEL folowed by a double quote and the label's name—in this case "LOOP. With Commodore LOGO, we designate a label by putting a colon behind the name of the label—for example, LOOP:

APPLE LOGO II
LOGO SYSTEMS LOGO — IBM PC & PCjr

```
TO GETCOMMAND
LABEL "LOOP
MAKE "INPUT RL
IF :INPUT = [] [PR [COMMAND?] GO "LOOP]
MAKE "VERB FIRST :INPUT
IF (COUNT :INPUT) = 1 [STOP]
MAKE "NOUN ITEM 2 :INPUT
IF (AND :NOUN = "THE (COUNT :INPUT) > 2) [MAKE "NOUN ITEM 3 :INPUT]
END
```

TERRAPIN LOGO — C-64

```
TO GETCOMMAND
LABEL "LOOP
MAKE "INPUT RQ
IF :INPUT = [] [PR [COMMAND?] GO "LOOP]
MAKE "VERB FIRST :INPUT
IF (COUNT :INPUT) = 1 [STOP]
MAKE "NOUN ITEM 2 :INPUT
IF (AND :NOUN = "THE (COUNT :INPUT) > 2) [MAKE "NOUN ITEM 3 :INPUT]
END
```

To use GO, we must first set up a label at the beginning of a loop. When LOGO encounters GO followed by that label at the end of the loop, it transfers control of the program back to the loop's starting point.

Normally, use of GO is avoided in LOGO; rather, we use *recursion* to re-execute a procedure. But recursion does not merely cause a procedure to loop back to its beginning, as many people think; rather, it causes a procedure to keep making and executing *copies* of itself at deeper and deeper levels until it encounters a reason to stop. We do not want GETCOMMAND to repeatedly generate clones of itself, therefore we will instead use GO for a simple loop back.

At the beginning of GETCOMMAND is a procedure (RECYCLE for Apple and IBM, .GCOLL for C-64) which forces a garbage collection so that used and discarded words and lists don't clutter up our workspace. Without this garbage collection, LOGO would eventually hold up our adventure for a lengthy garbage collection of its own. By placing this procedure at the top of GETCOMMAND we ensure that the process is done every time the loop is executed.

ID.LOC, the next procedure in GETCOMMAND, will give us information about our present location in our fantasy world. In Part 2 we will describe ID.LOC in full, but at this stage we will define it as a "dummy" procedure that will print out a general message.

APPLE LOGO II
LOGO SYSTEMS LOGO — IBM PC & PCjr

```
TO ID.LOC
PR []
PR [YOU ARE IN ONE OF MANY LOCATIONS]
END
```

TERRAPIN LOGO — C-64

```
TO ID.LOC
PR []
PR [YOU ARE IN ONE OF MANY LOCATIONS]
END
```

Next, GETCOMMAND will print out a prompt and use RQ (REQUEST on the C-64) or RL (READLIST for Apple and IBM) to assign whatever command is next typed in to a list named "INPUT. If :INPUT is empty—meaning the player hit the RETURN key without entering anything—the procedure will print EXCUSE ME . . . DID YOU SAY SOMETHING, and loop back to the top. If :INPUT is not empty, we execute PARSER.

APPLE LOGO II
LOGO SYSTEMS LOGO — IBM PC & PCjr

```
TO PARSER
MAKE "VERB
MAKE "NOUN
MAKE "VERB FIRST :INPUT
IF (COUNT :INPUT) = 1 [STOP]
MAKE "NOUN ITEM 2 :INPUT
IF (AND :NOUN = "THE (COUNT :INPUT) > 2) [MAKE "NOUN ITEM 3 :INPUT]
END
```

TERRAPIN LOGO — C-64

```
TO PARSER
MAKE "VERB
MAKE "NOUN
MAKE "VERB FIRST :INPUT
IF (COUNT :INPUT) = 1 [STOP]
MAKE "NOUN ITEM 2 :INPUT
IF (ALLOF :NOUN = "THE (COUNT :INPUT) > 2) [MAKE "NOUN ITEM 3 :INPUT]
END
```

The PARSER procedure is flexible enough to handle sentences like GIVE THE SCEPTRE TO THE PRINCESS or KILL THE SERPENTS WITH THE SWORD.

"Adventure game programs create the illusion that the computer understands English."

The first thing it does is inititalize two global variables, "VERB and "NOUN, by making them empty words.

Because we are dealing with commands only, the first word of a sentence will always be a verb. This simplifies things tremendously. We assign to the name "VERB the first word in the list :INPUT, and test to see if :INPUT was only one word long. If so, we return to GETCOMMAND.

If a command is longer than one word (like EXAMINE TREE or THROW THE JAR AT THE OGRE), we assume that the second word is a noun. We then test to see whether the article THE is the second word of the sentence, and whether the command is longer than two words. If both of these conditions are met, we assume that the third word is a noun. To conserve memory, we will not test for any adjectives—this omission will not affect the program. Now we are ready to return to GETCOMMAND.

At this point, GETCOMMAND tests to see if :VERB is a member of :VERBLIST, a predefined list of verbs that the program recognizes. If not, then a message is printed saying I DON'T KNOW HOW TO . . . and the program loops back up to the beginning of GETCOMMAND. If :VERB is recognized, :NOUN is checked to see whether it is EMPTY or if it is the word THE. If neither of these is true, it means that the program can use :VERB as a procedure and :NOUN as an input to that procedure. We'll look at how this works in a minute.

If :NOUN is empty or THE is returned from PARSER as the :NOUN, however, we either have a one-word command (such as N for North or I for Inventory), or there is no acceptable input to a two-word command. In the first case, we run the one-word command as a procedure. Otherwise, the only :NOUN we have is the word THE, so we need an error message asking for the actual :NOUN.

To determine whether the :VERB is a one- or two-word command, we divide :VERBLIST into two lists: :VERBLIST1, made up of one-word commands, and :VERBLIST2, which consists of verbs that require a :NOUN as input. If the :VERB is a member of :VERBLIST1, then the LIST primitive converts the lone :VERB into a list so that RUN will accept and execute it as a procedure. Here's a procedure, TO N, that will indicate a move to the North in our adventure game. For the time being, no real action is taken; instead, we will have the procedure respond with the message YOU WENT NORTH every time an N command is given.

APPLE LOGO II
LOGO SYSTEMS LOGO — IBM PC & PCjr

```
TO N↵
PR [YOU WENT NORTH]↵
END↵
```

TERRAPIN LOGO — C-64

```
TO N↵
PR [YOU WENT NORTH]↵
END↵
```

If the :VERB is not a member of :VERBLIST1, but *is* a member of :VERBLIST, then we have no input to a :VERB that requires a :NOUN. Trying to RUN a procedure that requires an input without including one causes the program to halt prematurely with an error message. To eliminate this potential error when this condition is discovered, we print the :VERB followed by WHAT?, and then return to the top of GETCOMMAND. For example, if the :INPUT is simply TAKE, then the procedure returns the message TAKE WHAT?

Eventually, these :VERB procedures will become quite complex; but for the time being we will simply return the statement YOU HAVE THE, followed by the :NOUN.

Finally, we return to the bottom of GETCOMMAND where the GO "LOOP takes us back to the beginning of GETCOMMAND, and we start looking for another command.

If you type in these procedures and try them out, you will find that halfway through GETCOMMAND you get the error message: THERE IS NO NAME VERBLIST. Here GETCOMMAND tests to see if :VERB is a member of :VERBLIST, which is set up by a procedure we will look at next time. To remedy this for now, type:

```
MAKE "VERBLIST1 (N)
MAKE "VERBLIST2 (TAKE)
MAKE "VERBLIST (N TAKE)
```

Now the program will understand either N or TAKE as part of :VERBLIST. Also, GETCOMMAND will know that TAKE requires a :NOUN as input, and that N requires no input.

"By the time we are done, we will have constructed an entire interactive fantasy in LOGO, and will have laid the groundwork for building countless other scenarios."

Now we'll look at how the program RUNs a :VERB as a procedure. This is a bit tricky, but it will reveal some valuable insights on how list processing works. Near the bottom of GETCOMMAND is the statement:

```
RUN (LIST :VERB WORD "" :NOUN)
```

Reading backwards from right to left, we see that WORD is used to glue quotes onto the value in :NOUN so that it is seen as a word rather than a procedure name. If we didn't do this, RUN would try to execute :NOUN as a procedure, instead of passing it along as an input to the procedure named by :VERB. Next, LIST turns :VERB and :NOUN into a list, for example, TAKE "SWORD, where TAKE is a procedure and "SWORD is the input to the procedure. RUN can then execute the following procedure:

APPLE LOGO II
LOGO SYSTEMS LOGO — IBM PC & PCjr

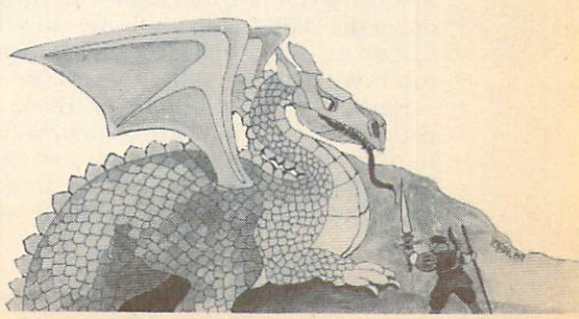
```
TO TAKE :OBJ↵
PR SE [YOU HAVE THE] :O
BJ↵
END↵
```

TERRAPIN LOGO — C-64

```
TO TAKE :OBJ↵
PR SE [YOU HAVE THE] :
OBJ↵↵
END↵
```

For any other word, it will respond with the I DON'T KNOW HOW TO message, followed by the unknown :VERB. You may experiment with this by adding to either "VERBLIST1 or "VERBLIST2 (always updating :VERBLIST) so that the program will recognize other commands. Make sure that you also define procedures to go with the new verbs.

In Part 2, we will discuss how to map out our fantasy world and convert it into data for the program to use. You may want to think about settings and locations that you would like to use. We will, of course, include our own for this particular scenario. Then, we will be able to use commands like N, S, E, and W for the compass directions, to walk around from place to place. **HCM**



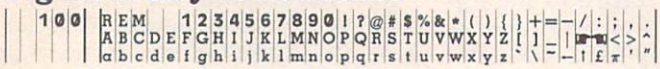
Your Guide to Typing in Programs from *HCM*

Within these pages is a software bonanza: entertainment, education, home and business applications, utilities, and tutorials—just for you. All you need to do is type them into your computer. *HCM* has taken most of the strain out of this process:

- Typeset listings with numbers in boldface.
- A bold, double vertical bar separating the line numbers from the program statements in BASIC listings.
- A vertical background grid to aid entry of the spaces.

Look at the Key-in-Reference (Figure 1 below) see how each character actually appears in the listing. By checking any questionable characters with the Key-in Reference, you can reduce errors to a minimum.

Figure 1: Key-in Reference



Before You Begin

Since *HCM* publishes for several different computers, the first thing you should do is make sure that you are looking at the listing designed for your machine. If, for example, you have an Apple IIe, make sure you look for the following black bar above the listing: APPLE II Family The computer model name will likewise appear on each subsequent page of each listing, so always look for the name before you begin typing from a new page of listings.

Before you begin typing in the program, you will want to set up a system to save your program. Whether you are using a cassette or diskette storage system, now is the time to be certain it is properly connected, powered up, and loaded with a blank cassette or an initialized disk. As you type in your program, you should get in the habit of saving your work after every twenty or so lines.

One of the most common errors in entering a listing is typing one symbol for another. These transpositions include substituting the letter O for the number 0, the letter l for the number 1, the letter S for the $, and the uppercase B for the number 8. The last error is especially likely when working in hexadecimal numbers which are composed of 0-9 and the uppercase letter A-F.

The listings in *HCM* are always the same number of characters wide, but the number of characters put on any line of the video display will vary from computer to computer. Don't try to make your listings *look like* the type-set listing—instead make sure you key in the listings character for character and space for space.

A Special Note on Listings: C-64 and VIC-20

Commodore uses more than 90 special symbols to represent various keyboard operations: for instance, the symbol [♥] in a program represents the operation of holding down the [SHIFT] key and pressing the key which has CLR on its upper half (second key from the right on the top row). This operation clears the screen.

Rather than reproducing these symbols, *HCM's* listings include key-stroke instructions, between two hands with pointing fingers. For example, when you find [SHIFT CLR] in an *HCM* listing, you will know to hold down the [SHIFT] key and press the key with CLR on it.

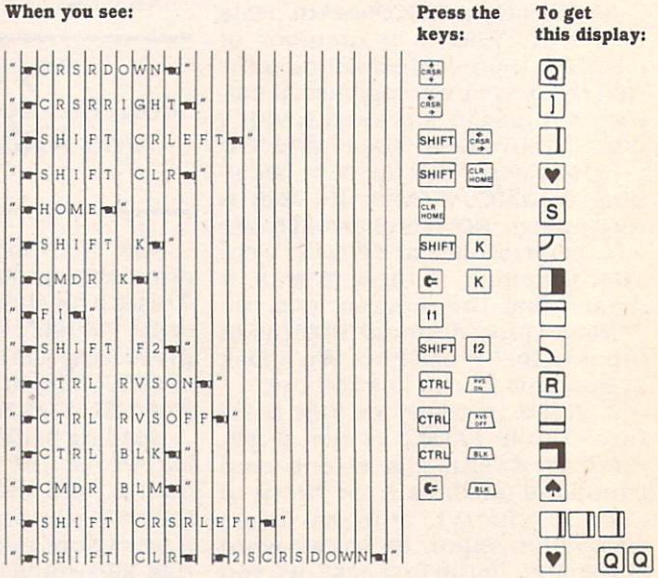
A number is included if you need to repeat the operation: [8SHIFT CRSRLEFT] tells you to hold the [SHIFT] key down and press the cursor left key (on the bottom right of the keyboard) eight times.

When you come to the hand symbols, remember:

- Each operation is enclosed in its own set of hand symbols.
- If any key action requires you to press two keys, press the control key or the Commodore key or the shift key *first* and hold it down before pressing the second key.
- Everything between a pair of hand symbols is set in a different place.

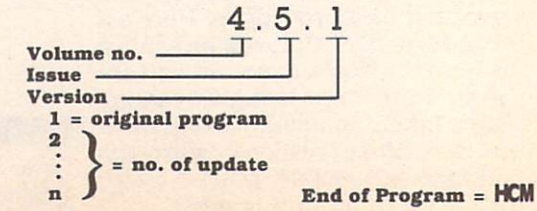
In Figure 2 below, we have included a chart showing you a representative sample of the symbols that appear when you use keystrokes enclosed by the hand symbols. (Notice that the hand symbols always appear within quotation marks—as in a print statement.)

Figure 2: Special symbols: C-64 and VIC-20



Program Identification

Each program header (the first few lines of the program) contains information giving the language the program is written in (e.g., TI Extended BASIC, Applesoft, etc.) and any special system components that are required (special memory cards, Speech Synthesizer, etc.). The first two digits of the version number tell you in which volume and issue of *HCM* the program *initially* appeared. The third digit of the version number indicates the version of the program. When a program initially appears, in *HCM*, it is version 1. Any subsequent revisions to the program if later published in the magazine or in the software available on magnetic medium from *HCM* will bear a revised version number.



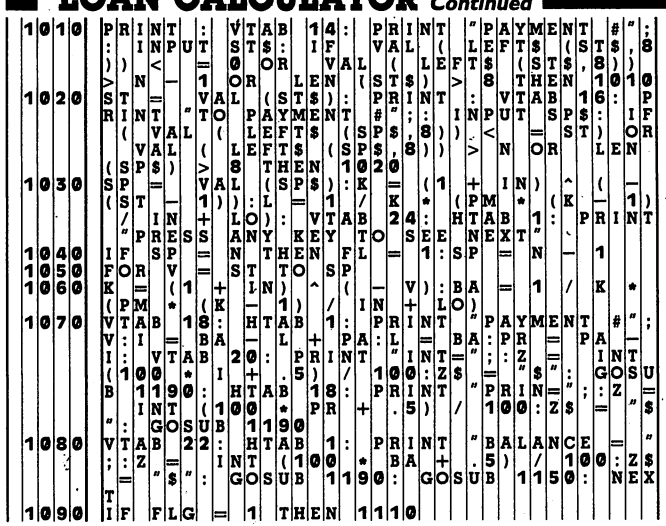
CONTENTS

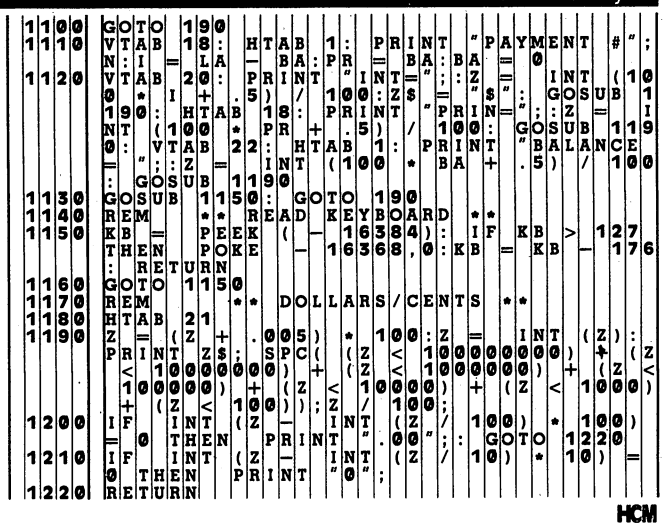
LOAN CALCULATOR

Continued

LOAN CALCULATOR *Continued*

APPLE // Family

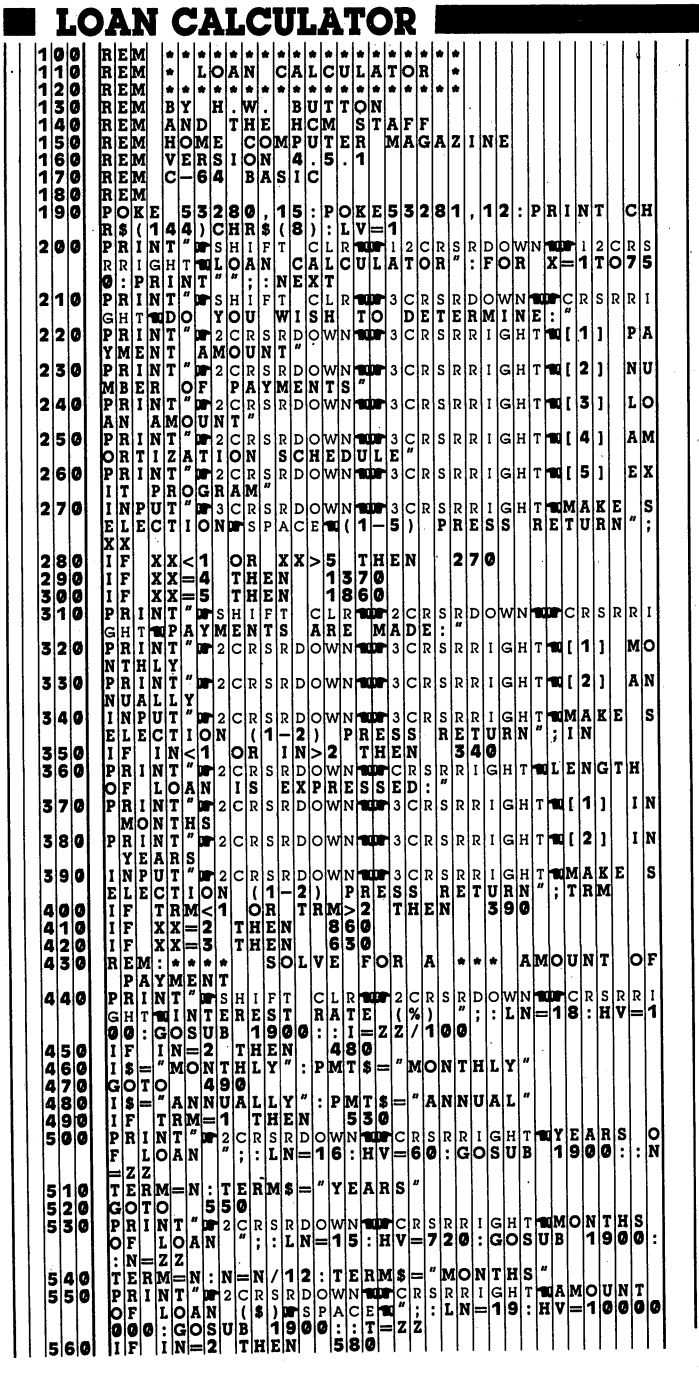


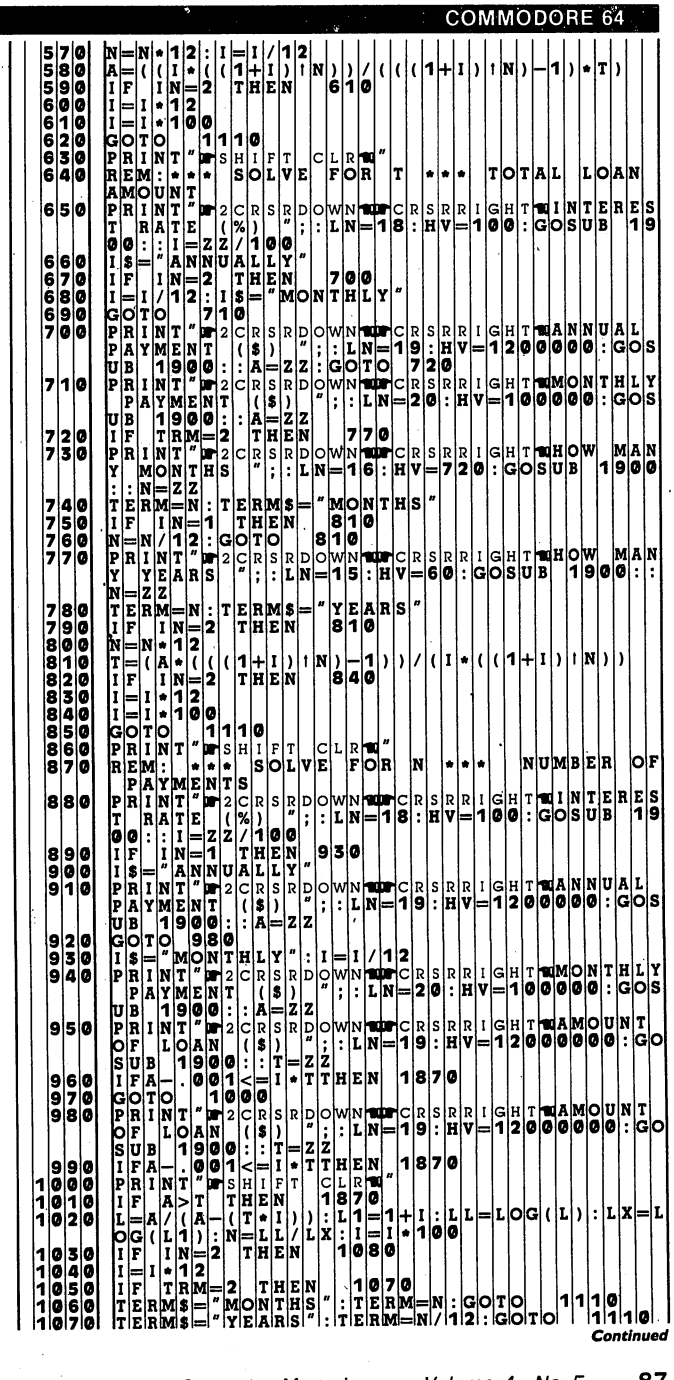


HCM

LOAN CALCULATOR

COMMODORE 64





Continued

PROGRAM LISTING

LOAN CALCULATOR *Continued*

COMMODORE 64

```
1080 IF TRM=1 THEN 1100
1090 TERM$="YEARS":TERM=N:GOTO 1110
1100 TERM$="MONTHS":TERM=N*12
1110 PRINT"{SHIFT CLR}"
1120 REM ** CALCULATE *** FINAL REPORT
1130 PRINT"{CRSR DOWN}{CRSR RIGHT}INTEREST RATE: ";I;"%"
1140 PRINT"{CRSR DOWN}{CRSR RIGHT}COMPOUNDED ";I$
1150 T$=STR$((INT((T+.005)*100))/100)
1160 C$="0":E=VAL(RIGHT$(T$,2))
1170 IF E<1 AND E>0 THEN T$=T$+C$
1180 PRINT"{CRSR DOWN}{CRSR RIGHT}LOAN AMOUNT $";T$
1190 A$=STR$(INT((A+.005)*100)/100):C$="0":B=VAL(RIGHT$(A$,2))
1200 IF B<1 AND B>0 THEN A$=A$+C$
1210 IF IN<>1 THEN PRINT"{CRSR DOWN}{CRSR RIGHT}ANNUAL PAYMENT $";A$:GOTO 1240
1220 REM
1230 PRINT"{CRSR DOWN}{CRSR RIGHT}MONTHLY PAYMENTS $";A$
1240 PRINT"{CRSR DOWN}{CRSR RIGHT}NO OF PAYMENTS ";STR$(INT((N+.005)*100)/100)
1250 TM$="TERM IN "+TERM$:PRINT"{CRSR DOWN}{CRSR RIGHT}";TM$;" ";STR$(INT((TERM+.005)*100)/100)
1260 TP=INT((A*N+.005)*100)/100
1270 TP$=STR$(TP):C$="0":B=VAL(RIGHT$(TP$,2))
1280 IF B<1 AND B>0 THEN TP$=TP$+C$
1290 PRINT"{CRSR DOWN}{CRSR RIGHT}TOTAL COST $";TP$
1300 IT=INT((TP-T+.005)*100)/100:IT$=STR$(IT):D= VAL(RIGHT$(IT$,2))
1310 IF D<1 AND D>0 THEN IT$=IT$+C$
1320 PRINT"{CRSR DOWN}{CRSR RIGHT}TOTAL INTEREST $";IT$
1330 PRINT"{3 CRSR DOWN}{CRSR RIGHT}TO CONTINUE PRESS ANY KEY EXCEPT RUN/STOP"
1340 GET Y$:IF Y$=""THEN 1340
1350 Z$="[5] END SESSION"
1360 GOTO 210
1370 REM: **** AMORTIZATION SCHEDULE *****
1380 PRINT"{SHIFT CLR}{CRSR DOWN}{CRSR RIGHT}LOAN AMOUNT ($)";:LN=16:HV=12000000:GOSUB 1900::LOAN=ZZ
1390 PRINT"{CRSR DOWN}{CRSR RIGHT}NO OF MONTHLY PAYMENTS";:LN=22:HV=720:GOSUB 1900::N=ZZ
1400 PRINT"{CRSR DOWN}{CRSR RIGHT}INTEREST RATE (%)";:LN=18:HV=100:GOSUB 1900::IN=ZZ/1200
1410 PMT=LOAN*(IN/(1-(1+IN)↑(-N)))
1420 ALL=INT(PMT*N*100)/100
1430 PAY=INT(PMT*100)/100
1440 LASTP=ALL-PAY*(N-1)
1450 PA$=STR$(PAY):C$="0":B=VAL(RIGHT$(PA$,2))
1460 IF B<1 AND B>0 THEN PA$=PA$+C$
1470 PRINT"{CRSR DOWN}{CRSR RIGHT}MONTHLY PAYMENT= $";PA$
1480 LA$=STR$(INT((LASTP+.005)*100)/100):C$="0":D=VAL(RIGHT$(LA$,2))
1490 IF D<1 AND D>0 THEN LA$=LA$+C$
1500 PRINT"{CRSR DOWN}{CRSR RIGHT}FINAL PAYMENT= $";LA$
1510 PRINT"{2 CRSR DOWN}{CRSR RIGHT}SHOW SCHEDULE FROM PAYMENT #";:LN=28:HV=N:GOSUB 1900::SR=ZZ
1520 PRINT"{CRSR DOWN}{CRSR RIGHT}END SCHEDULE WITH PAYMENT #";:LN=27:HV=N:GOSUB 1900::SP=ZZ
1530 K=(1+IN)↑(-(SR-1)):L=1/K*(PMT*(K-1)/IN+LOAN)
1540 IF SP<N THEN FLG=0
1550 IF SP=N THEN FLG=1:SP=N-1
1560 FOR Z=SR TO SP
1570 K=(1+IN)↑(-Z):BAL=1/K*(PMT*(K-1)/IN+LOAN)
1580 PRINT"{SHIFT CLR}{8 CRSR DOWN}{14 CRSR RIGHT}PAYMENT #";Z
1590 I=BAL-L+PAY:L=BAL:PR=PAY-I
1600 II=INT((I+.005)*100)/100:PP=INT((PR+.005)*100)/100
1610 BB=INT((BAL+.005)*100)/100:II$=STR$(II):C$="0":D=VAL(RIGHT$(II$,2))
1620 IF D<1 AND D>0 THEN II$=II$+C$
1630 PP$=STR$(PP):C$="0":B=VAL(RIGHT$(PP$,2))
1640 IF B<1 AND B>0 THEN PP$=PP$+C$
1650 BB$=STR$(BB):C$="0":D= VAL(RIGHT$(BB$,2))
1660 IF D<1 AND D>0 THEN BB$=BB$+C$
1670 PRINT"{3 CRSR DOWN}{CRSR RIGHT}INT.= $";II$,"PRIN.=$";PP$
1680 PRINT"{CRSR DOWN}{CRSR RIGHT}BALANCE=$";BB$
1690 PRINT"{4 CRSR DOWN}{CRSR RIGHT}TO CONTINUE PRESS ANY KEY EXCEPT RUN/STOP"
1700 GET Y$: IF Y$=""THEN 1700
1710 NEXT Z
1720 IF FLG=0 THEN 1850
1730 IF SR=N THEN 1850
1740 PRINT"{SHIFT CLR}{8 CRSR DOWN}{14 CRSR RIGHT}PAYMENT #";N
1750 I=LASTP-BAL:PR=BAL:BAL=0
1760 II=INT((I+.005)*100)/100:PP=INT((PR+.005)*100)/100
1770 II$=STR$(II):C$="0":D=VAL(RIGHT$(II$,2))
1780 IF D<1 AND D>0 THEN II$=II$+C$
1790 PP$=STR$(PP):C$="0":B=VAL(RIGHT$(PP$,2))
1800 IF B<1 AND B>0 THEN PP$=PP$+C$
1810 PRINT"{3 CRSR DOWN}{CRSR RIGHT}INT.= $";II$,"PRIN.= $";PP$
1820 PRINT"{CRSR DOWN}{CRSR RIGHT}BALANCE= $";INT((BAL+.005)*100)/100
1830 PRINT"{4 CRSR DOWN}{CRSR RIGHT}TO CONTINUE PRESS ANY KEY EXCEPT RUN/STOP"
1840 GET Y$: IF Y$=""THEN 1840
1850 GOTO 210
1860 PRINT"{SHIFT CLR}":END
1870 PRINT"{3 CRSR DOWN}{CRSR RIGHT}THE CALCULATIONS CANNOT BE MADE BASED UPON THE DATA INPUT.";
1880 PRINT" PLEASE WAIT!"
1890 FOR X=1TO1750:PRINT"";:NEXT:GOTO 210
1900 TB$="{40 CRSR RIGHT}"
1910 SP$="                                        "
1920 INPUT "";ZZ$:IF ZZ$="" OR ZZ$=CHR$(13) THEN 2020
1930 IF LEN(ZZ$)>10 THEN 2020
1940 FORZZ=1TOLEN(ZZ$):IFASC(MID$(ZZ$,ZZ,1))<48 THEN 1970
1950 IFASC(MID$(ZZ$,ZZ,1))>57THEN 1970
1960 GOTO 1980
1970 IFASC(MID$(ZZ$,ZZ,1))<>46THEN 2020
1980 NEXT
1990 IFVAL(ZZ$)>HVTHEN 2020
2000 IFVAL(ZZ$)<LVTHEN 2020
2010 ZZ=VAL(ZZ$):RETURN
2020 PRINT"{SHIFT CRSR UP}";LEFT$(TB$,LN+1);LEFT$(SP$,38-LN):PRINT"{SHIFT CRSR UP}";LEFT$(TB$,LN+1);
2030 GOTO 1920
```

HCM

LOAN CALCULATOR

IBM PC & IBM PCjr

```
100 '   ******************
110 '   * LOAN CALCULATOR *
120 '   ******************
130 '   BY H. W. BUTTON
140 '   AND THE HCM STAFF
150 '   HOME COMPUTER MAGAZINE
160 '   VERSION 4.5.1
170 '   IBM PCjr WITH CARTRIDGE BASIC OR
180 '   IBM PC WITH BASICA
190 '
200 '   INITIALIZE PROGRAM
210 SCREEN 0:CLS:KEY 15,CHR$(&H0)+CHR$(&H1):KEY 16,CHR$(&H40)+CHR$(&H1):ON KEY(15) GOSUB 1250:ON KEY(16) GOSUB 1250:KEY OFF:F$="$$###,###,###,###.##":F2$="###########.####":IST$="########.######   %"
220 ON ERROR GOTO 1230:DEFDBL A-Y:SCRN=0:KEY(15) ON:KEY(16) ON:LOCATE 12,12:PRINT "LOAN CALCULATOR":LOCATE 22,1:PRINT "PRESS ";CHR$(17);CHR$(196);CHR$(217);" TO START -- [Esc] TO EXIT"
230 A$=INKEY$:IF A$<>CHR$(13) THEN 230
240 '
250 '  DISPLAY MAIN MENU SCREEN
260 CLS:SCRN=1:LOCATE 1,12:PRINT "LOAN CALCULATOR":LOCATE 5,1:PRINT "PRESS A NUMBER TO SELECT AN OPTION:"
270 LOCATE 7,6:PRINT "1) PAYMENT AMOUNT":PRINT:PRINT "2) NUMBER OF PAYMENTS":PRINT:PRINT "3) AMOUNT OF LOAN":PRINT:PRINT "4) AMORTIZATION SCHEDULE":PRINT:PRINT "5) EXIT PROGRAM":LOCATE 20,1:PRINT "SELECT ONE:"
280 A$=INKEY$:IF A$<"1" OR A$>"5" THEN 280 ELSE OPT=VAL(A$)
290 SCRN=2:ON OPT GOTO 320,320,320,910,1270
300 '
310 '  PAYMENT TYPE FOR OPTIONS 1 TO 3
320 CLS:LOCATE 1,1:PRINT "PAYMENTS ARE MADE:":LOCATE 3,6:PRINT "1) MONTHLY":PRINT:PRINT "2) ANNUALLY"
330 A$=INKEY$:IF A$<"1" OR A$>"2" THEN 330 ELSE IN=VAL(A$):IF IN=1 THEN I$="MONTHLY":PMT$=I$ ELSE I$="ANNUALLY":PMT$="ANNUAL"
```

Continued

LOAN CALCULATOR *Continued*

IBM PC & IBM PCjr

```
340 LOCATE 10,1:PRINT "LENGTH OF LOAN IN
    S EXPRESSED
    "1) MONTHS":PRINT:PRINT "
    YEARS"
350 A$=INKEY$:IF A$<>"1" OR A$>"2" THEN
    350 ELSE TRM=VAL(A$):CLS:ON TRM GOT
    O 480,680,580
360 '
370 ' ROUTINES TO GET INFORMATION
380 ' ALL OPTIONS
390 PRINT "INTEREST RATE:";:GOSUB 116
    0:I=VAL(A$)/100:RETURN
400 IF TRM=1 THEN PRINT "MONTHS OF LOANS
    ";:RM$="MONTHS" ELSE:RM$="YEARS";
    :PRINT "YEARS OF LOAN:";
410 GOSUB 1160:N=VAL(A$):RETURN
420 PRINT "AMOUNT";:GOSUB 1160:
    P=VAL(A$):RETURN
430 IF I=1 THEN PRINT "MONTHLY PAYMENT
    ";:GOSUB 1160:RETURN ELSE:PRINT "ANNUAL PAYMENT";:GOSUB 1160
440 R=VAL(A$):RETURN
450 '
460 ' AMOUNT OF PAYMENTS ROUTINE
470 ' OPTION #1
480 LOCATE 1,11:PRINT "AMOUNT OF PAYMEN
    TS"
490 LOCATE 3,1:GOSUB 390
500 LOCATE 5,1:GOSUB 400:IF TRM=1 THEN
    N=N/12 ELSE LTERM=N
510 LOCATE 7,1:GOSUB 420
520 IF TRM=2 THEN LTERM=N*12 ELSE LTERM
    =N
530 I=I:IF I=1 THEN N=N*12:I=I/12
540 A=(I*((1+I)^N))/(((1+I)^N)-1):T=I
550 '
560 ' NUMBER OF PAYMENTS ROUTINE
570 ' OPTION #2
580 LOCATE 1,13:PRINT "AMOUNT OF LOAN"
590 LOCATE 3,1:GOSUB 390
600 LOCATE 5,1:GOSUB 430
610 LOCATE 7,1:GOSUB 400:IF TRM=1 THEN
    N=N/12
620 IF TRM=2 THEN LTERM=N ELSE LTERM=N
    *12
630 I=I:IF I=1 THEN N=N*12:I=I/12
640 T=(A*(((1+I)^N)-1))/(I*((1+I)^N)):I
    =I*100:GOTO 770
650 '
660 ' AMOUNT OF LOAN ROUTINE
670 ' OPTION #3
680 LOCATE 1,11:PRINT "NUMBER OF PAYMEN
    TS"
690 LOCATE 3,1:GOSUB 390
700 LOCATE 5,1:GOSUB 420
710 LOCATE 7,1:GOSUB 430
720 I=I:IF I=1 THEN I=I/12
730 IF I*P*(I+1)^N/((1+I)^N-1)<A THEN Z
    =Z+1:GOTO 730 ELSE LOCATE 2,1:PRIN
    T "PAYMENT DOES
    NOT EXCEED LOAN
    ";:FOR T=1 TO 2000:NEXT:CLS:GOTO 6
    80
740 N=LOG(A/(A-(T*I)))/LOG(1+I):I=I*100
    :IF I<=0 THEN PRINT "INTEREST RATE
    IS TOO LOW":END ELSE IF TRM=2 THEN
    N=N*12
750 '
760 ' REPORT FOR OPTIONS 1 TO 3
770 CLS:LOCATE 1,14:PRINT "LOAN REPORT"
780 LOCATE 3,1:PRINT USING "INTEREST RAT
    E IS ##.###%";I
790 LOCATE 5,1:PRINT "COMPOUNDED ";:IF
    I=1 THEN PRINT "MONTHLY" ELSE PRINT
    "ANNUALLY"
800 LOCATE 7,1:PRINT "LOAN AMOUNT ";:PRI
    NT USING "$$#,###,###.##";P
810 LOCATE 9,1:PRINT "PAYMENT ";:PRINT U
    SING "$$#,###,###.##";A
820 LOCATE 11,1:PRINT "NO. OF PAYMENTS"
    ;:PRINT USING "####";N
830 LOCATE 13,1:PRINT USING "TERM ####
    ";LTERM;RM$
840 LOCATE 15,1:PRINT "TOTAL COST ";:PR
    INT USING "$$#,###,###.##";A*N
850 LOCATE 17,1:PRINT "INTEREST ";:PRIN
    T USING "$$#,###,###.##";(A*N)-P
860 LOCATE 22,1:PRINT "PRESS ";:CHR$(17)
    ;CHR$(196);CHR$(217);" TO CONTINUE"
870 A$=INKEY$:IF A$="" THEN 870
880 '
890 ' AMORTIZATION SCHEDULE ROUTINE
900 '
910 CLS:PRINT "AMORTIZATION #";:INPUT N
    :CLS:LOCATE 1,13:PRINT "AMORTIZATIO
    N SCHEDULE"
920 LOCATE 3,1:GOSUB 390:I=I/12
930 LOCATE 5,1:GOSUB 400:IF TRM=2 THEN
    N=N/12ELSE
940 LOCATE 7,1:GOSUB 420
950 P=T*(I/(1-(1+I)^(-N))):TOT=INT(P*N*
    100+.5)/100:PAY=INT(P*100+.5)/100:
    LAST=TOT-PAY*(N-1)
960 CLS:LOCATE 1,10:PRINT "AMORTIZATION
    REPORT"
970 LOCATE 3,1:PRINT "INTEREST RATE";I*
    100;"%":PRINT USING "LOAN AMOUNT $$
    #,###,###.##";P
980 LOCATE 5,1:PRINT USING "MONTHLY PAY
    MENT $$#,###,###.##";PAY
990 LOCATE 7,1:PRINT USING "FINAL PAYME
    NT $$#,###,###.##";LAST
1000 LOCATE 9,1:PRINT "SHOW SCHEDULE #"
     ;:GOSUB 1160:FROM=A:IF FROM=0 OR
     FROM>N-1 THEN 1000
1010 TO=FROM+11:IF TO>N THEN TO=N
1020 K=(1+I)^(FROM-1):L=1/K*(P*T*(K-1)
     /I+T)
1030 FOR Z=FROM TO TO:K=(1+I)^(-Z):BAL=
     1/K*(P*T*(K-1)/I+T)
1040 LOCATE 16,12:PRINT "PAYMENT #";Z
1050 I=CBAT(BAL-L+PAY):PRINT USING "PRI
     NCIPAL $$#,###,###.##";I
1060 LOCATE 18,1:PRINT "INTEREST $";TAB
     (19);:PRINT USING "$$#,###,###.##
     ";PAY-I:PRINT "REMAINING BALANCE
     $";:PRINT USING "$$#,###,###.##";
     BAL
1070 LOCATE 25,1:PRINT "PRESS ";:CHR$(17
     );CHR$(196);CHR$(217);" TO CONTINU
     E"
1080 Z$=INKEY$:IF Z$<>CHR$(13) THEN 108
     0
1090 NEXT:GOTO 260
1100 '
1110 '*********************
1120 '* USER INTERFACE ROUTINES *
1130 '*********************
1140 '
1150 ' NUMERIC KEY SCANNING ROUTINE
1160 X=POS(0):A$="":DEF SEG=0:P=PEEK(10
     50):X=PEEK(1052):XPA=XP:LOCATE XP,
     YP:PRINT "";:A$=INKEY$:IF A$="" THE
     N 1160 ELSE IF A$=CHR$(13) THEN 117
     0
1170 K=INSTR("0123456789.",A$):IF K=0 TH
     EN 1160 ELSE IF K=11 AND INSTR(A$,"
     .")>0 THEN 1160
1180 PRINT A$;:IF LEN(A$)>0 THEN P=P*10
     +K-1:XP=XP+1:POKE 1050,PEEK(1052)+
     1:GOTO 1160
1190 IF K=11 THEN P=P/100:XP=XP+2:POKE 1
     050,PEEK(1052)+2:GOTO 1160
1200 '
1210 '
1220 '
1230 FOR I=1 TO 10:LOCATE 25,1:PRINT "
     ";:SOUND 110,1:FOR T=1 TO 100:NEXT
     :LOCATE 25,1:PRINT "PRESS [Esc] KE
     Y THEROUTINE";:SOUND 220,1:FOR T=1
     TO 100:NEXT:NEXT
1240 '
1250 IF N=2 THEN RETURN 260 ELSE IF SCRN
     =1 THEN RETURN 210 ELSE IF SCRN=0
     THEN RETURN 1270 ELSE RETURN
1260 '
1270 CLS:PRINT "SEE YOU NEXT TIME":END
```

HCM

LOAN CALCULATOR

TI-99/4A

```
100 REM ***********************
110 REM * LOAN CALCULATOR *
120 REM ***********************
130 REM BY H.W. BUTTON
140 REM HOME COMPUTER MAGAZINE
150 REM VERSION 4.5.1
160 REM EXTENDED BASIC
170 REM
180 ON ERROR 1870 :: ON WARNING NEXT
190 CALL CLEAR
200 DISPLAY AT(10,7):"LOAN CALCULATOR"
    :::: FOR DELAY=1 TO 500 :: NEXT DELAY
    :::: CALL CLEAR
210 CALL CLEAR
220 DISPLAY AT(3,1):"DO YOU WISH TO DET
    ERMINE:"
230 DISPLAY AT(6,3):"1) PAYMENT AMOUNT"
    :::: DISPLAY AT(8,3):"2) NUMBER OF
    PAYMENTS":::: DISPLAY AT(10,3):"3)
    AMOUNT OF LOAN"
240 DISPLAY AT(12,3):"4) AMORTIZATION S
    CHEDULE":::: DISPLAY AT(14,3):"5)
    EXIT PROGRAM";
250 CALL KEY(0,X,Y):: IF Y<>1 THEN 250
260 IF X<49 OR X>53 THEN 250
270 IF X=53 THEN 1540
280 IF X=52 THEN 1540
290 CALL CLEAR :: CALL LOAD("DSK1.LOAN
    1")
300 DISPLAY AT(3,1):"PAYMENTS ARE MADE:
    ":::: "1 MONTHLY":"2 ANNUALLY"
```

Continued

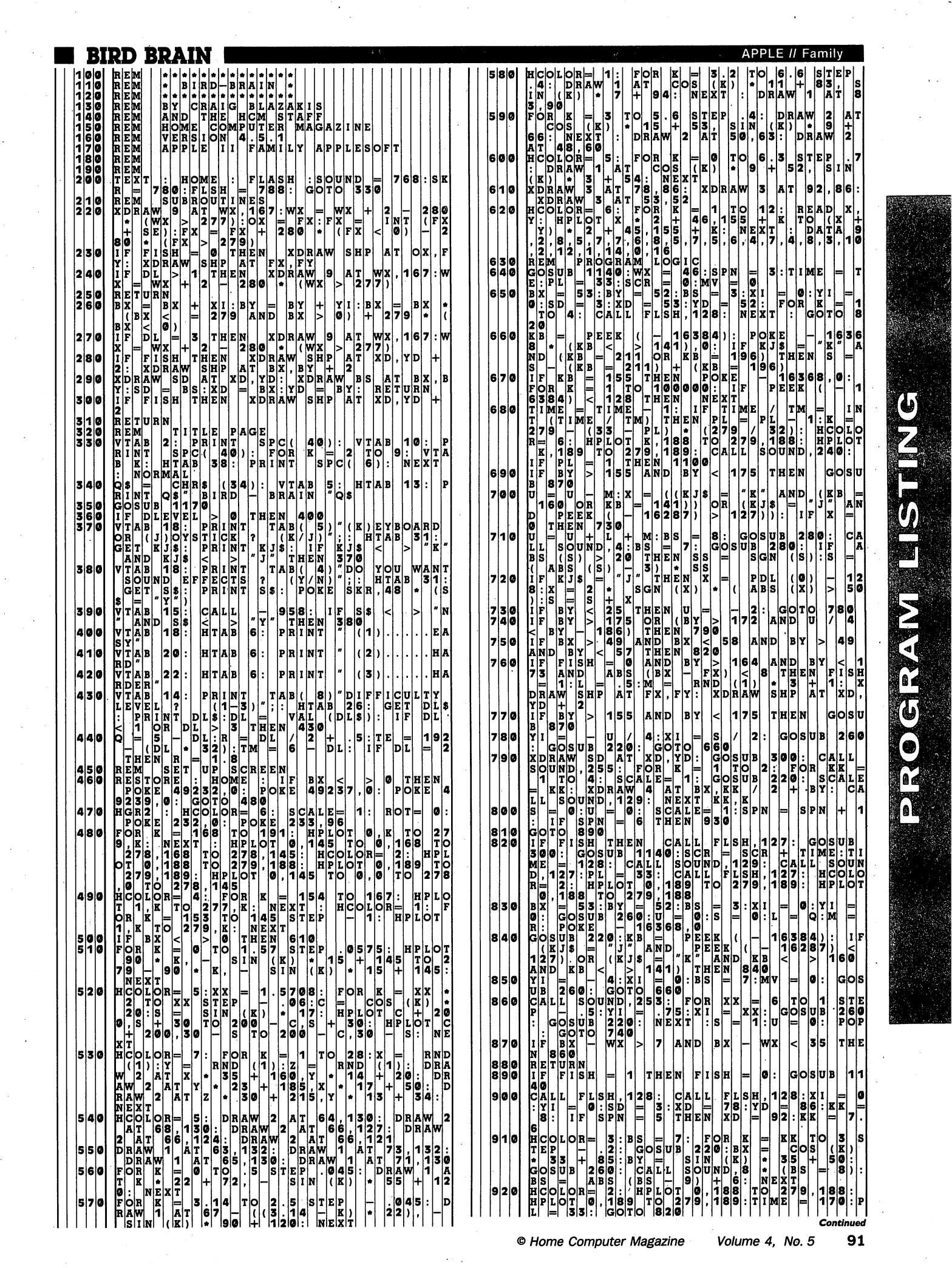
PROGRAM LISTING

LOAN CALCULATOR *Continued*

PROGRAM LISTING

```
310 CALL KEY(0,INTEREST,Y)::IF INTERES
    T<49 OR INTEREST>50 OR Y=-1 THEN 31
    0
320 CALL CLEAR
330 DISPLAY AT(3,1):"LENGTH OF LOAN IS
    EXPRESSED:::DISPLAY AT(7,8):"1 - 2
    IN MONTHS:::DISPLAY AT(10,8):"BEEP
    :::"2 - IN YEARS"::::CALL KEY(0,TRM,Y)::IF TRM<49 OR TR
    M>50 OR Y=-1 THEN 340
340 CALL CLEAR
350 IF XX=50 THEN 1020
360 IF XX=51 THEN 690
370 REM **SOLVE FOR A***                                AMO
    UNT OF PAYMENT
380 INPUT "INTEREST RATE (%) ":I
390 PRINT
400 I=I/100
410 IF INTEREST=50 THEN 460
420 I$="MONTHLY"
430 PMT$="MONTHLY"
440 GOTO 480
450 I$="ANNUALLY"
460 PMT$="ANNUALLY"
470 IF TRM=49 THEN 540
480 INPUT "YEARS OF LOAN ":N
490 PRINT
500 TERM=N
510 TERM$="YEARS"
520 GOTO 590
530 INPUT "MONTHS OF LOAN ":N
540 PRINT
550 TERM=N
560 N=N/12
570 TERM$="MONTHS"
580 INPUT "AMOUNT OF LOAN $":T
590 CALL CLEAR
600 IF INTEREST=50 THEN 640
610 N=N*12
620 I=I/12
630 A=(I*((1+I)^N))/(((1+I)^N)-1)*T
640 IF INTEREST=50 THEN 670
650 I=I*100
660 GOTO 1400
670 CALL CLEAR
680 REM **SOLVE FOR T**                                TOTA
    L LOAN AMOUNT
690 INPUT "INTEREST RATE (%) ":I
700 PRINT
710 I=I/100
720 I$="ANNUALLY"
730 IF INTEREST=50 THEN 790
740 I=I/12
750 I$="MONTHLY"
760 GOTO 820
770 INPUT "ANNUAL PAYMENT $":A
780 GOTO 830
790 PRINT "MONTHLY PAYMENT $":A
800 IF TRM=50 THEN 910
810 INPUT "HOW MANY MONTHS? ":N
820 TERM=N
830 TERM$="MONTHS"
840 IF INTEREST=49 THEN 960
850 N=N/12
860 GOTO 960
870 INPUT "HOW MANY YEARS? ":N
880 TERM=N
890 TERM$="YEARS"
900 IF INTEREST=50 THEN 960
910 N=N*12
920 T=(A*(((1+I)^N)-1))/(I*((1+I)^N))
930 IF I=50 THEN 1000
940 I=I*12
950 I=I*100
960 GOTO 1400
970 CALL CLEAR
980 REM **SOLVE FOR NUMB
    ER OF PAYMENTS
990 INPUT "INTEREST RATE (%) ":I
1000 I=I/100
1010 IF INTEREST=49 THEN 1110
1020 I$="ANNUAL PAYMENT $":A
1030 GOTO 1150
1040 I$="MONTHLY"
1050 I=I/12
1060 PRINT "MONTHLY PAYMENT $":A
1070 PRINT "AMOUNT OF LOAN $":T
1080 CALL CLEAR
1090 L=A/(A-(T*I))
1100 L=L-1
1110 L=LOG(L)
1120 L=L/LOG(1+I)
1130 N=INT(L)
1140 IF N=INT(L) THEN 1330
1150 I=I*12
1260 IF TRM=50 THEN 1300
1270 TERM$="MONTHS"
1280 TERM=N
1290 GOTO 1400
1300 TERM$="YEARS"
1310 TERM=N/12
1320 GOTO 1400
1330 IF TRM=49 THEN 1370
1340 TERM$="YEARS"
1350 TERM=N
1360 GOTO 1400
1370 TERM$="MONTHS"
1380 TERM=N*12
1390 REM **CALCULATE**                                  F
    INAL REPORT
1400 CALL CLEAR
1410 DISPLAY AT(3,1):USING 1850:I :::DIS
    PLAY AT(5,1):"COMPOUNDED";I$
1420 T=(INT((T*100)))/100
1430 DISPLAY AT(7,1):USING 1840:"LOAN AM
    OUNT",T
1440 IF INTEREST<>49 THEN DISPLAY AT(9,1
    ):USING 1840:"ANNUAL PAYMENT",A::
    GOTO 1460
1450 DISPLAY AT(9,1):USING 1840:"MONTHLY
    PAYMENT",A
1460 DISPLAY AT(11,1):USING 1860:"NO. OF
    PAYMENTS",N
1470 TM$="TERM IN "&TERM$:::DISPLAY AT(
    13,1):USING 1860:TM$,TERM
1480 TP=A*N::DISPLAY AT(15,1):USING 18
    40:"TOTAL COST",TP
1490 TI=TP-T
1500 TI=TP-T::::DISPLAY AT(17,1):USING 1
    840:"INTEREST",TI::::DISPLAY AT(24,1):"TO CONTINUE PRESS ANY KEY
    "
1510 CALL KEY(0,X,Y)::IF Y<>1 THEN 1510
1520 CZ$="5-END SESSION"
1530 GOTO 210
1540 END
1550 CALL CLEAR
1560 REM ++AMORTIZATION
1570 CHEDULE++
    DISPLAY AT(2,1):"LOAN AMOUNT? $":::
    DISPLAY AT(2,16):BEEP:LOAN::::DISPLA
    Y AT(4,1):"NO OF MONTHLY PAYMENTS?
    ::ACCEPT AT(4,25)BEEP:N
1580 DISPLAY AT(6,1):"INTEREST RATE? (%)
    ":::IN
    ACCEPT AT(6,20)BEEP:IN
1590 IN=IN/1200
1600 PMT=LOAN*(IN/(1-(1+IN)^(-N))))
1610 TOTAL=INT(PMT*N*100)/100
1620 PAST=INT(PMT*100)/100
1630 PLAST=TOTAL-PAY*(N-1)
1640 DISPLAY AT(8,1):"MONTHLY PAYMENT =
    $";PAY
1650 DISPLAY AT(10,1):"FINAL PAYMENT =
    $";PLAST
1660 DISPLAY AT(12,1):"SHOW SCHEDULE FRO
    M BEGINNING PAYMENT #":::ACCEPT AT(14,10
    )BEEP:STRT
1670 DISPLAY AT(16,1):"TO PAYMENT #":::
    ACCEPT AT(16,13)BEEP:STP
1680 K=(1+IN)^(-(STRT-1))::L=1/K*(PMT*(
    K-1)/IN+LOAN)::DISPLAY AT(24,1):"PR
    ESS ANY KEY TO SEE NEXT";;
1690 IF STP=N THEN FLG=1::STP=N-1 ELSE
    FLG=0
1700 FOR Z=STRT TO STP
1710 K=(1+IN)^(-Z)::BAL=1/K*(PMT*(K-1)/
    IN+LOAN)
1720 DISPLAY AT(18,1)BEEP:"PAYMENT #";Z
1730 I=BAL-L+PAY::L=BAL::PR=PAY-I
1740 DISPLAY AT(20,1):"INT=$";INT(100*I+
    .5)/100,"PRIN=$";INT(100*PR+.5)/100
1750 DISPLAY AT(22,1):"BALANCE = $";INT(
    100*BAL+.5)/100
1760 CALL KEY(0,K,S)::IF S<>1 THEN 1760
1770 NEXT Z
1780 IF FLG=0 THEN 1830
1790 DISPLAY AT(18,1)BEEP:"PAYMENT #";N
1800 PR=BAL::BAL=0
1810 DISPLAY AT(20,1):"INT=$";INT(100*I+
    .5)/100,"PRIN=$";INT(100*PR+.5)/100
1820 DISPLAY AT(22,1):"BALANCE = $";INT(
    100*BAL+.5)/100
1830 CALL KEY(0,K,S)::IF S<>1 THEN 1830
1840 IMAGE ########.##
1850 IMAGE INTEREST RATE:  ##.###%
1860 IMAGE ##########.##
1870 FOR ER=1 TO 3::DISPLAY AT(24,1):"
    ERROR IN CALCULATION";::CALL SOUND
    (150,110,0)::FOR TD=1 TO 200::NE
    XT TD
1880 DISPLAY AT(24,1):"FOR ER=1 TO 100
    ::NEXT ER::RETURN 1800
```

HCM



Continued

BIRD BRAIN *Continued*

APPLE // Family

```
930 REM  END OF GAME & SCORE
940 HOME : T EXT : HOME
950 FLASH: K = 2 TO 4: VTAB K: PRI
    NT SPC( 40);: NEXT
    RMAL: PRINT : PRINT "DO YOU WA
    NT TO PLAY AGAIN ? A$";
960 VTAB 9: PRINT TAB( 5)"ACCUMULATED
    TIME FOR "; TAB( 5)"LEFT EACH
    ENTRY:"; TAB( 30);"DIFFICULTY
970 VTAB 13: PRINT TAB( 5)"DIFFICULTY
    LEVEL 1-4: HTAB 25: PRINT
980 VTAB 18: HTAB 30: PRINT HSCR: NORMA
    L
990 PRINT TAB( 5)"TOTAL SCORE = ";
    TAB( 30);SCR
1000 VTAB 18: PRINT TAB( 5)"HIGH SCORE
    ";
1010 INVERSE : VTAB 1: PRINT SPC( 40): P
    RINT SPC( 40);: VTAB 22: PRINT
    SPC( 40);: VTAB 23: PRINT SPC(
    40);: VTAB 22: NEXT
1020 IF HSCR AND SCR > 0 THEN HSCR = SCR
    : IF HSCR = SCR THEN FLASH
1030 VTAB 18: HTAB 30: PRINT HSCR: NORMA
    L
1040 IF SCR * DL = HSCR AND SCR > 0 THEN
    FOR K = 1 TO 8: CALL SOUND, 129:
    CALL SOUND, 1127: NEXT
1050 VTAB 21: HTAB 4: PRINT "DO YOU WANT
    TO PLAY AGAIN ? (Y/N) ";: HTAB 31:
    GET A$: PRINT A$
1060 IF A$ < > "N" THEN A$ < > "Y" AND
    A$ = "Y" THEN 100
1070 HOME : IF A$ = "Y" THEN 200
1080 END
1090 REM
1100 REM  TIME OUT
1110 CALL SOUND, 1: CALL SOUND, 255: GOSUB
    CALL SOUND, 65: CALL SOUND, 300:
    IF BX = 277 THEN BX = 277
1120 FOR K = BY TO 12 STEP 4: XDRAW 4 AT
    BX, K: XDRAW 8 AT BX, K: SOUND
    = 8: XDRAW 8 AT BX, K: NEXT
    CALL FLSH, 127: CALL SOUND, 65:
    CALL FLSH, 127: SPN
```

```
1130 GOTO 890
1140 FX = 1: FY = 1: FISH = 0: RN = INT
    ( RND (1) * 3 + 1: IF RN = 1 THEN
    SHP = 5: XDRAW 1 AT 169: OX = 2)
    RN: SHP = 6: XDRAW 1 AT OX, FY:
    RETURN
1150 SHP = FXD: FY = RETURN: SHAPES
    SHP AT FXD, FY: RETURN
1160 REM  POKE DATA & RETURN
1170 IF STO = 768 THEN 1220
1180 RESTORE : FOR K = 768 TO 811: READ P:
    POKE K, P: NEXT
1190 FOR K = 768 TO 811: READ P: POKE K,
    P: NEXT
1200 DATA 11,11,32,76,231,193,34,6,169,0,
    168,136,208,32,76,231,142,193,56,3,
    229,6,208,244,96,32,76,231,142,39,
    3,160,0,132,38,165,32,76,231,142,
    62,177,38,73,127,145,38,200,208,247,
    230,32,177,38,201,127,240,242,96
1210 DATA 32,76,231,142,39,3,160,0,132,38,
    165,32,76,231,142,62,177,38,73,127,
    145,38,200,208,247,230,32,177,38,
    201,127,240,242,96
1220 IF FOR = 1 THEN 1250
1230 FOR K = 768 TO 811: READ P: POKE K,
    P: NEXT
1240 FOR K = 768 TO 811: READ P: POKE K,
    P: NEXT
1250 RESTORE : RETURN
1260 REM  DATA
1270 DATA 9,20,0,20,0,31,0,56,0,67,0,80,0,
    96,0,39,1,45,0,7,0,39,0,0,168,0,54,
    46,0,45,37,0,65,34,54,46,0,36,37,
    63,0,54,54,37,36,44,54,37,36,44,
    36,60,39,63,63,0,44,60,36,36,53,
    54,37,54,43,13,5,0,45,32,59,63,23,
    14,45,0,170,2,62,59,63,133,0,54,
    45,45,45,45,45,63,63,37,39,63,55,
    39,37,41,53,6,0,37,36,60,37,37,
    37,63,62,62,60,60,60,60,54,22,9,
    9,9,13,33,32,37,37,41,37,62,53,
    37,53,53,54,0,37,36,60,54,62,9,
    9,9,63,54,18,18,18,9,9,37,44,9,
    36,9,50,30,27,51,27,46,17,13,42,
    17,13,42,17,5,0
1280 DATA 37,44,44,44,44,53,6,0,12,33,9,
    12,33,9,17,36,9,36,9,12,33,9,12,
    33,9,17,13,42,17,13,42,17,5,0
1290 REM
1300 REM
1310 REM
```

HCM

BIRD BRAIN

COMMODORE 64

```
100 REM  ********** BRAIN **********
110 REM  *  BIRD  BRAIN  *
120 REM  ********** BLAZAKIS **********
130 REM  BY CRAIG BLAZAKIS
140 REM  AND THE HCM STAFF
150 REM  HOME COMPUTER MAGAZINE
160 REM  VERSION 4.5.1
170 REM  C-64 BASIC
180 REM
190 REM
200 POKE 52,48:POKE 56,48:POKE 55,0:CLR:FO
    RZ=53248TO53263:POKEZ,0:NEXT
210 GOTO 2320
220 POKESE,PEEK(SE)AND251:POKEYM(2),0:P
    OKEXM(2),0:POKECM,2:POKEX(2),15:GOT
    O400
230 X=PEEK(X(0)):Y=PEEK(Y(0)):EN=PEEK(S
    E)AND1
240 IF(X=XN)AND(Y=YN)AND(EN=1)THEN390
250 POKEMB,PEEK(MB)AND254
260 IF(PEEK(SE)AND48)=0THEN2960
270 IF(PEEK(SE)AND16)=0THEN300
280 POKEX(0),PEEK(X(4)):POKEY(0),PEEK(Y
    (4))
290 POKESE,PEEK(SE)AND239:GOTO310
300 POKEX(0),PEEK(X(5)):POKEY(0),PEEK(Y
    (5)):POKESE,PEEK(SE)AND223
310 POKESP(0),254:POKESE,PEEK(SE)OR1:PO
    KEXM(0),0:POKEYM(0),253
320 POKESM,PEEK(SM)OR1:POKECM,1
330 IFPEEK(Y(0))>YN+10THENGOSUB490:GOTO
    330
340 POKEXM(0),253:POKEYM(0),250:POKECM,
    1
350 IFPEEK(X(0))>XN+10THENGOSUB490:GOTO
    350
360 POKEXM(0),0:POKEX(0),XN:POKEYM(0),2
    :POKECM,1
370 IFPEEK(Y(0))<YN-3THEN370
380 POKEYM(0),0:POKEY(0),YN:POKESP(0),2
    54
390 FT=1:FO=0:UV=0:SV=0:RETURN
400 POKEY(2),201:POKEY(2),201:N=INT(RND
    (1)*6)+1:ONNGOTO410,420,430,440,450
    ,460
410 POKECS(2),7:POKESP(2),249:FC=1:POKE
    XM(2),3:GOTO470
420 POKECS(2),7:POKESP(2),250:FC=1:POKE
    XM(2),253:GOTO470
430 POKECS(2),15:POKESP(2),249:FC=1:POK
    EXM(2),3:GOTO470
440 POKECS(2),15:POKESP(2),250:FC=1:POK
    EXM(2),253:GOTO470
450 POKECS(2),13:POKESP(2),249:FC=1:POK
    EXM(2),3:GOTO470
460 POKECS(2),13:POKESP(2),250:FC=1:POK
    EXM(2),0:POKEXM(2),253:POKEYM(2),0:
    POKEX(2),XN:POKEY(2),YN:POKECM,2:PO
    KESP(2),249:POKESE,PEEK(SE)OR4:POKE
    CM,2:ET=344:LT=344:POKEM,0
470 GOSUB500:POKESM,PEEK(SM)OR4:POKESE,
    PEEK(SE)OR4:POKECM,2:ET=344:LT=344:
    POKEM,0
480 GOTO230
490 POKESP(0),252:FORI=1TO400:NEXT:POKE
    SP(0),251:RETURN
500 POKEX(6),88:POKEMB,PEEK(MB)OR64:POK
    EY(6),242:POKECS(6),6
510 POKESE,PEEK(SE)OR64
520 PRINT"■HOME■ 24 CRSR DOWN■CTRL RVSON
    ■CTRL 39 SHIFT C■";
530 POKE56,295,6:POKE2023,192:T0=TI:T1=0
    :POKEXM(0),0:POKEMB,PEEK(MB)AND254
540 RETURN
550 T2=INT((TI-T0)/TL):T3=T2-T1:T1=T2:I
    F(T3<1THEN640
560 ET=PEEK(X(6)):IF(PEEK(MB)AND64)<>0T
    HEN=ET+256
570 ET=ET-T3
580 IFET<LT-8THEN610
590 LT=ET-8:IFLT=24THEN1270
600 LPOKE(LT-24)/8+1984),160
610 IFET<256THENPOKEMB,PEEK(MB)AND191:G
    OTO630
620 ET=ET-256:POKEMB,PEEK(MB)OR64
630 POKEX(6),ET
640 RETURN
650 REM
660 IFFT=0THENGOSUB550
670 IFP$="K"THENGETD$:FORZZ=1TO10:GETZZ
    $:NEXTZZ:GOTO720
680 K=PEEK(J1)AND12:FF=PEEK(J1)AND16:FF
    =ABS(FF-16)
690 IFK=8THEND$="S":GOTO720
700 IFK=4THEND$="D":GOTO720
710 D$="":F(PEEK(MB)AND1)=0THENGOTO740
720 IF(PEEK(X(0)))>60)AND(PEEK(X(0)))<100
    )THENPOKEX(0),1:POKEMB,PEEK(MB)AND2
    54
730 IF(PEEK(MB)AND2)=0THENGOTO760
740 IF(PEEK(X(1)))>60)AND(PEEK(X(1)))<100
    )THENPOKEX(1),1:POKEMB,PEEK(MB)AND2
    53
750 F(PEEK(MB)AND4)=0THENGOTO780
760 IF(PEEK(X(2)))>60)AND(PEEK(X(2)))<100
    )THENPOKEX(2),1:POKEMB,PEEK(MB)AND2
    51
770 IFP$="K"THENGETD$:FORZZ=1TO10:GETZZ
    $:NEXTZZ:GOTO720
780 IFPEEK(X(0))<240THEN800
790 IF(PEEK(MB)AND1)=1THENPOKEX(0),56
800 IFPEEK(X(1))<240THEN820
810 IF(PEEK(MB)AND2)=2THENPOKEX(1),56
```

Continued

```
 820 IFPEEK(X(2))<240THEN840
 830 IF(PEEK(MB)AND4)=4THENPOKEX(2),56
 840 IFFT=0THEN880
 850 FO=0:IF(D$="")AND(FF=0)THEN650
 860 FT=0:POKEYM(0),247:POKEXM(0),0:POKE
     SM,PEEK(SM)OR1:POKECM,1:HV=0
 870 FO=0:GOSUB500
 880 V=PEEK(YM(0)):Y=PEEK(Y(0))
 890 IF(D$<>"")OR(FF=16)THENPOKESP(0),25
     2:GOTO970
 900 X=PEEK(X(0)):Y=PEEK(Y(0)):IF(X>XN+1
     0)OR(X<XN-10)THEN950
 910 IF(Y>YN+2)OR(Y<YN-2)THEN950
 920 M=PEEK(MB)AND1:IFM=1THEN950
 930 H=0:HV=0:IFFO=1THEN1430
 940 POKEXM(0),0:POKEYM(0),0:POKEY(0),YN
     :UV=0:SV=0:POKESP(0),254:GOTO650
 950 IFFO<>0THENUV=UV-(FC+FS):GOTO1050
 960 UV=UV-FS:GOTO1050
 970 IFY<51THENPOKEY(0),50:UV=0:GOTO1000
 980 IFFO<>0THENUV=UV+LC:GOTO1000
 990 UV=UV+LS
1000 IFD$="S"THENHV=HV-1:IFHV<(-8)THENHV
     =(-8)
1010 IFD$="D"THENHV=HV+1:IFHV>8THENHV=8
1020 IFHV<0THENH=246-HV:GOTO1050
1030 IFHV>0THENH=10-HV:GOTO1050
1040 H=0
1050 IFY<50THENUV=4:V=10
1060 IFUV<8THEN1080
1070 IFV<255THENV=V+1:GOTO1100
1080 IFUV=>-3THEN1110
1090 IFV>1THENV=V-1
1100 UV=0
1110 IFV=0THENV=247
1120 IFV>128ANDV<246THENV=10:GOTO1140
1130 IFV<127ANDV>10THENV=246
1140 POKEYM(0),V:POKEXM(0),H:IFFO=1THENP
     OKEXM(2),H:POKEYM(2),V
1150 POKECM,1:POKESP(0),251:IFFO=0THEN11
     80
1160 QX=PEEK(X(0)):QY=PEEK(Y(0)):PX=PEEK
     (X(2)):PY=PEEK(Y(2))
1170 IFABS(QX-PX)>5ORABS(QY-PY)>5THENPOK
     EX(2),PEEK(X(0)):POKEY(2),PEEK(Y(0)
     )
1180 IFPEEK(Y(0))>215THEN1370
1190 I=PEEK(53278)AND7:IF(I=0)OR(I=1)OR(
     I=2)OR(I=4)OR(I=6)THEN650
1200 IF((I=3)OR(I=7))AND(PEEK(Y(0))>178)
     THEN1240
1210 IF(I<>5)ORY<190THEN650
1220 POKEXM(2),H:POKEYM(2),V:POKECM,1:PO
     KEX(2),PEEK(X(0)):POKEY(2),PEEK(Y(0
     ))
1230 FO=1:GOTO650
1240 POKEXM(0),1:POKEYM(0),8:UV=8:IFFO=1
     THENPOKEYM(2),8:POKEXM(2),1
1250 POKECM,1:GOTO650
1260 REM  HEAVEN BOUND
1270 E=PEEK(SE):E=E+8:IFFO<>0THENE=E-2
1280 POKEXM(0),0:POKEYM(0),0:POKEX(3),PE
     EK(X(0)):POKEY(3),PEEK(Y(0))
1290 POKESM,6
1300 POKESP(3),253:POKESP(0),254
1310 POKECM,1:M=PEEK(MB):IF(MANDM)=1THEN
     M=MOR8:GOTO1330
1320 M=MANDM247
1330 POKEMB,M:POKESE,E:POKEYM(0),247:POK
     EYM(3),247:POKEXM(3),0:POKEYM(3),24
     7
1340 POKESM,15:POKECM,1:FO=0
1350 IFPEEK(Y(0))<30THENPOKESE,(E-9):GOS
     UB220:GOTO650
1360 GOTO1350
1370 POKESM,PEEK(SM)AND254:POKEXM(0),0:P
     OKEYM(0),0:POKEXM(2),0:POKEYM(2),0
1380 POKECM,1
1390 POKESP(0),248:POKECS(0),1:FORI=1TO1
     000:NEXT
1400 POKESE,PEEK(SE)AND254:POKECS(0),9
1410 IFFO=1THENPOKESM,PEEK(SM)AND251:FO=
     0
1420 GOSUB220:GOTO650
1430 POKEXM(0),0:POKEYM(0),0:POKEXM(2),0
     :POKEYM(2),0:POKESE,PEEK(SE)AND251:
     POKEY(0),YN
1450 POKEX(0),XN:GOSUB220:FORZ=1TO20:GET
     D$:NEXT:HV=0:FO=0:GOTO650
1460 POKESM,PEEK(SM)AND251
1470 FORI=15808TO16319:READA:POKEI,A:NEX
     T
1480 DATA0,3,136,0,0,15,192,0,0,0,31
1490 DATA226,0,63,240,0,127,255,240,0
1500 DATA255,255,136,0,3,255,0,3,255,10
1510 DATA1,255,0,3,255,0,3,255
1520 DATA0,7,255,0,7,255,0,0,15
1530 DATA255,128,15,255,128,31,255,192
1540 DATA31,255,224,63,255,224,63,255
1550 DATA240,127,255,248,255,255,252,89
1560 DATA0,0,0,0,0,0,0,0
1570 DATA0,1,128,0,96,78,0,16
1580 DATA72,16,8,72,16,4,72,32
1590 DATA50,72,64,74,72,128,132,1
1600 DATA6,2,0,120,56,0,0,79
1610 DATA147,62,2,16,32,32,129,16,72,64
1620 DATA0,136,32,0,0,152,0,0
1630 DATA32,0,0,0,0,0,0,0
1640 DATA0,0,0,0,0,0,0,0
1650 DATA0,0,0,0,0,0,0,0
1660 DATA0,0,0,0,0,0,0,0
1670 DATA0,16,0,0,0,24,0,0,0,28
1680 DATA0,8,63,0,0,12,127,192,14
1690 DATA255,0,16,0,0,15,255,240,7,255,240
1700 DATA0,0,0,0,0,0,0,0
1710 DATA0,0,0,0,0,0,0,0
1720 DATA0,0,0,0,0,0,0,0
1730 DATA0,0,0,0,0,0,0,0
1740 DATA0,0,0,0,0,0,0,0
1750 DATA0,8,0,0,0,24,0,0,0,56
1760 DATA0,0,252,16,3,252,48,5
1770 DATA255,0,112,0,15,255,240,15,255,224
1780 DATA0,0,0,0,0,0,0,0
1790 DATA0,0,0,0,0,0,0,0
1800 DATA0,0,0,0,0,0,0,0
1810 DATA0,0,8,0,0,20,0,0,0
1820 DATA28,0,14,8,56,49,221,198
1830 DATA64,6,32,1,0,28,0,0,28
1840 DATA0,0,34,0,0,0,34,0,0,0
1850 DATA85,0,0,0,0,0,0,0,0
1860 DATA0,0,0,0,0,0,0,0
1870 DATA0,0,0,0,28,0,0,28,98,0
1880 DATA35,1,28,64,0,148,0,128,0
1890 DATA156,120,0,73,0,0,93,0
1900 DATA0,62,0,28,0,0,0,28
1910 DATA0,0,34,0,0,0,34,0,0,0
1920 DATA85,0,0,0,0,0,0,0,0
1930 DATA0,0,0,0,0,0,0,0
1940 DATA0,0,0,0,0,0,0,0
1950 DATA0,58,192,0,0,239,176,0,250
1960 DATA240,3,255,255,240,255,3,255,252,15
1970 DATA255,252,255,15,255,252,3,255,255,252
1980 DATA3,255,252,252,0,255,255,0,255
1990 DATA240,3,253,255,0,3,127,215,13
2000 DATA127,85,253,87,85,213,85,85
2010 DATA213,85,87,85,85,87,85,85
2020 DATA85,213,85,85,0,0,61,127,87,223
2030 DATA0,0,0,0,0,0,0,0
2040 DATA0,8,0,0,0,0,0,0,0
2050 DATA0,0,0,0,0,0,0,0
2060 DATA28,0,0,0,0,0,28,0,0
2070 DATA0,62,0,0,0,28,0,0,28
2080 DATA0,0,34,0,0,0,34,0,0
2090 DATA85,0,0,0,0,0,0,0,0
2100 DATA0,0,0,0,0,0,0,0
2110 DATA0,0,0,0,0,0,0,0
2120 RETURN
2130 REM  AUTO SPRITE MOTION
2140 FORI=50880TO51116:READA:POKEI,A:NEXT
2150 DATA169,255,45,0,198,240,16,169,0,1
     41,0,198,162,21,189,0,
2160 DATA197,157,0,198,202,208,247,162,1
     ,169,1,141,80,197,173,80,
2170 DATA197,45,80,197,240,3,76,243,198,2
     32,232,14,80,197,208,238
2180 DATA76,49,234
2190 DATA169,0,29,0,197,208,3,76,97,199,
     169,128,61
2200 DATA0,197,240,48,254,0,198,208,40,2
     22,255,207,76,144,199,80
2210 DATA197,45,16,208,208,12,173,16,208
     ,13,80,197,141,16,208,76,
2220 DATA43,199,173,16,208,77,80,197,141
     ,16,208,189,0,197,157,0,
2230 DATA198,76,97,199,222,0,198,208,40,
     254,255,207,208,29,173,80
2240 DATA197,45,16,208,208,12,173,16,208
     ,13,80,197,141,16,208,76,
2250 DATA91,199,173,16,208,77,80,197,141
     ,16,208,189,0,197,157,0,
2260 DATA198,169,0,232,29,0,197,208,3,76
     ,140,199,169,128,61,0,
2270 DATA197,240,11,254,0,198,208,20,222
     ,255,207,76,134,199,222,0,
2280 DATA198,208,9,254,255,207,189,0,197
     ,157,0,198,202,76,233,198
2290 DATA169,255,221,255,207,240,3,76,43
     ,199,173,80,197,76,17,199
2300 DATA120,169,192,141,20,3,169,198,14
     ,21,3,88,96
2310 RETURN
2320 FORI=0TO14STEP2:A=I/2:X(A)=53248+I:
     Y(A)=53249+I:XM(A)=50433+I
2330 YM(A)=50434+I:NEXT
2340 FORI=16320TO16382:POKEI,255:NEXT:PO
     KE16383,0
2350 FORI=0TO7:CS(I)=53287+I:SP(I)=2040+
     I:NEXT:HS=0
2360 POKEYM(6),0:POKESP(6),255
2370 GOSUB23800:GOTO2470
2380 POKE53285,3:POKE53286,7:POKECS(3),4
2390 POKECS(0),9:POKECS(1),6:POKECS(4),9
     :POKECS(5),9:SE=53269:MB=53264
2400 SC=53281:BD=53280:POKE53277,247:POK
     E53276,8
2410 POKESP(1),247:POKEX(1),32:POKEY(1),
     196:POKEYM(1),0:SM=50432:CM=50688
2420 JI=56320:J2=56321:POKEX(2),150:POKE
     Y(2),201:POKEYM(2),0
```

Continued

PROGRAM LISTING

BIRD BRAIN *Continued* — COMMODORE 64

```
2430 POKEX(4),130:POKEX(5),152:POKEY(4),130:POKEY(5),140
2440 POKESP(0),254:POKESP(4),254:POKESP(5),254
2450 XN=60:YN=94:FT=1:PT=0:POKEX(0),XN:POKEY(0),YN:POKESM,0
2460 RETURN
2470 PRINT"{SHIFT CLR}{6 CRSR DOWN}":POKE53280,5:POKE53281,1:POKE646,5:POKESE,0
2480 PRINT"{5 CRSR RIGHT}****************************"
2490 PRINT"{5 CRSR RIGHT}****************************"
2500 PRINT"{5 CRSR RIGHT}**                        **"
2510 PRINT"{5 CRSR RIGHT}**                        **"
2520 PRINT"{5 CRSR RIGHT}**                        **"
2530 PRINT"{5 CRSR RIGHT}**    B I R D   B R A I N **"
2540 PRINT"{5 CRSR RIGHT}**                        **"
2550 PRINT"{5 CRSR RIGHT}**                        **"
2560 PRINT"{5 CRSR RIGHT}**                        **"
2570 PRINT"{5 CRSR RIGHT}****************************"
2580 PRINT"{5 CRSR RIGHT}****************************"
2590 GOSUB1470
2600 PRINTSPC(6)"{5 CRSR DOWN}";
2610 PRINT"JOYSTICK OR KEYBOARD? (J/K)";
2620 GETP$:IFP$<>"J"ANDP$<>"K"THEN2620:PRINTP$
2630 GOSUB2640:GOTO2700
2640 PRINT"{SHIFT CLR}{12 CRSR DOWN}{8 CRSR RIGHT}1 = EASY"
2650 PRINT"{CRSR DOWN}{8 CRSR RIGHT}2 = HARD"
2660 PRINT"{CRSR DOWN}{8 CRSR RIGHT}3 = HARDER"
2670 PRINT"{HOME}{8 CRSR DOWN}{5 CRSR RIGHT}":PRINT"DIFFICULTY LEVEL (1-3)";
2680 GETL$:IFL$<"1"ORL$>"3"THEN2680
2690 L=VAL(L$):PRINT"{SHIFT CLR}":RETURN
2700 GOSUB2130
2710 H=0:HV=0:ONLGOTO2720,2730,2740
2720 TL=40:LS=4:FS=2:LC=2:WS=4:GOTO2750
2730 TL=32:LS=3:FS=3:LC=1.6:WS=3:GOTO2750
2740 TL=24:LS=2:FS=4:LC=1.3:WS=2
2750 PRINT"{SHIFT CLR}{4 CRSR DOWN}"SPC(33)"{CTRL CYN}{CMDR P}{2 CMDR O}{CMDR I}{CTRL RVSON}{CMDR U}{CTRL RVSOFF}{CMDR I}{CMDR P}";
2760 PRINTSPC(33)"{CMDR T}{CMDR Y}{CMDR U}{CMDR Y}{CMDR T}"
2770 PRINT"{2 CRSR DOWN}"SPC(5)"{CMDR BLK}{CMDR *}{CTRL RVSON} {CTRL RVSOFF}{SHIFT £}{CTRL CYN}"TAB(23)"{CMDR @}{CMDR P}{2 CMDR O}{CMDR I}{2 CMDR O}{CMDR P}{CMDR @}"
2780 PRINTSPC(3)"{CTRL GRN}{CTRL RVSON}{SHIFT £} {CMDR *}{CTRL RVSOFF}{CTRL CYN}"TAB(23)"{2 CMDR T}{2 CMDR Y}{2 CMDR T}"
2790 PRINT"   {CTRL RVSON}{CTRL GRN} {CTRL RVSOFF} {CMDR @}{CMDR P}{CMDR O}{CMDR P}{CMDR @}"
2800 PRINT"   {CTRL RVSON}{CTRL GRN} {CTRL RVSOFF}{CMDR K}{CTRL RVSON}{CMDR J} {CMDR WHT} {CTRL GRN} {CMDR J}{CTRL RVSOFF} {CTRL RVSON}{SHIFT £} {CMDR *}"SPC(10)"{CTRL CYN}{CTRL RVSOFF}{CMDR P}{2 CMDR O}{CMDR I}{CMDR O}{CMDR P}{CMDR @}"
2810 PRINT"   {CTRL RVSON}{CTRL GRN} {CTRL RVSOFF}{CMDR J}{CTRL RVSON}{CMDR K} {CMDR WHT} {CTRL GRN} {CMDR K}{CTRL RVSOFF} {CTRL RVSON}"
2820 PRINT"{CTRL RVSOFF}  {CTRL RVSON}{CTRL GRN} {CTRL RVSOFF}{CMDR H}{CMDR N}{CTRL RVSON} {CMDR WHT} {CTRL GRN}{CTRL RVSOFF}{CMDR L}{CTRL RVSON} {CTRL RVSOFF} {CTRL RVSON} {CMDR WHT} {CTRL GRN} {CTRL RVSOFF}{CMDR K}{CTRL RVSON}{CMDR H} {CTRL RVSOFF}"
2830 PRINT"   {CTRL RVSON}{CTRL GRN} {CTRL RVSOFF} {CTRL RVSON} {CMDR WHT} {CTRL RVSOFF} {CTRL RVSON}{CTRL GRN} {CTRL RVSOFF} {CTRL RVSON}{CMDR WHT}{SHIFT £}{CTRL GRN} {CMDR WHT} {CTRL RVSOFF}{SHIFT £}{CTRL GRN}{CTRL RVSON}{CMDR J} {CTRL RVSOFF}{CMDR J}{CTRL RVSON}{CMDR J} {CTRL RVSOFF}{CMDR WHT}"
2840 PRINT"   {CTRL GRN}{SHIFT £} {CMDR WHT}{CTRL RVSON} {CTRL RVSOFF} {CTRL GRN}{CMDR *}{CMDR WHT}{CTRL RVSON}{SHIFT £} {CTRL GRN} {CTRL RVSOFF} {CTRL RVSON}{CMDR K} {CTRL RVSOFF}{CMDR H}{CTRL RVSON}{CMDR K} {CTRL RVSOFF}"
2850 PRINTSPC(6)"{CMDR WHT}{CTRL RVSON} {CTRL RVSOFF} {CTRL RVSON}{SHIFT £} {CTRL RVSOFF}{SHIFT £} {CTRL RVSON}{CTRL GRN} {CTRL RVSOFF}"
2860 PRINT"{CTRL YEL}{CMDR I}{CMDR O}{CMDR P}{CMDR @} {CMDR WHT}{CTRL RVSON} {CRSR RIGHT}{SHIFT £} {CTRL RVSOFF}{SHIFT £}"
2870 PRINT"{CTRL YEL}{CTRL RVSON}   {CMDR Y}{CMDR WHT} {SHIFT £} {CTRL RVSOFF}{SHIFT £}"SPC(12)"{CTRL YEL}{CMDR @}{CMDR P}{CMDR O}{CMDR I}{CTRL RVSON}{CMDR U} {CMDR Y}{CMDR U}{CTRL RVSOFF}{CMDR I}{CMDR O}{CMDR P}{CMDR @}"
2880 PRINT"{CTRL RVSON}{CTRL YEL}  {CMDR WHT} {CTRL YEL}{CTRL RVSOFF}{2 CMDR I}{3 CMDR O}{3 CMDR P}{CMDR O}{CMDR I}{CTRL RVSON}{CMDR U}{CMDR T}";
2890 PRINT"{CTRL YEL}{CTRL RVSON}          ";
2900 FORI=1TO4:PRINT"{CTRL BLU}{CTRL RVSON}              ";:NEXT
2910 GOSUB500:POKESE,63
2920 POKESE,PEEK(SE)OR2:POKESM,PEEK(SM)OR2:POKEXM(1),WS:SYS51104
2930 POKESE,51:POKEXM(4),0:POKEYM(4),0:POKEXM(5),0:POKEYM(5),0
2940 GOSUB220:POKEXM(0),0
2950 FT=1:GOTO650
2960 POKESE,0
2970 PRINT"{CTRL GRN}{CTRL RVSOFF}{SHIFT CLR}{5 CRSR DOWN}{5 CRSR RIGHT}ACCUMULATED TIME"
2980 PRINT"{5 CRSR RIGHT}LEFT ON CLOCK FOR"
2990 PRINT"{5 CRSR RIGHT}EACH FISH EATEN";PT
3000 PRINT"{CRSR DOWN}{5 CRSR RIGHT}DIFFICULTY LEVEL      X";L
3010 PRINTSPC(25)"{5 CMDR T}"
3020 PRINT"{5 CRSR RIGHT}TOTAL SCORE"SPC(9)PT*L
3030 IFPT*L>HSTHENHS=PT*L
3040 PRINT"{3 CRSR DOWN}{5 CRSR RIGHT}HIGH SCORE"SPC(10)HS
3050 PRINT"{5 CRSR DOWN}{5 CRSR RIGHT}"
3060 POKE198,0
3070 INPUT"PLAY AGAIN? (Y/N)";Z$
3080 IFZ$<>"Y"ANDZ$<>"N"THEN3070
3090 IFZ$="Y"THENGOSUB2380:GOSUB2640:GOTO2710
3100 PRINT"{SHIFT CLR}{3 CRSR DOWN}{5 CRSR RIGHT}THANK YOU{2 CRSR DOWN}{4 SHIFT CRSR LEFT}AND{2 CRSR DOWN}{SHIFT CRSR LEFT}COME AGAIN"
3110 END
```

HCM

BIRD BRAIN — IBM PC & IBM PCjr

```
100 ' **************
110 ' * BIRD-BRAIN *
120 ' **************
130 ' BY CRAIG BLAZAKIS
140 ' HOME COMPUTER MAGAZINE
150 ' VERSION 4.5.1
160 ' IBM PCjr CARTRIDGE BASIC
      FROM DOS 2.1
170 ' IBM PC BASICA WITH
180 ' COLOR GRAPHICS MONITOR ADAPTER
190 ' AND COLOR MONITOR
200 SCREEN 0:KEY OFF:WIDTH 40:COLOR 3,1,1:CLS:RANDOMIZE TIMER:STRIG ON:PLAY "MF"
210 LOCATE 12,10:PRINT "B I R D - B R A I N":LOCATE 23,7:PRINT "PRESS [ENTER] TO CONTINUE"
220 A$=INKEY$:IF A$="" THEN 220 ELSE CLS:SCREEN 1:COLOR 0,0
230 DEFINT B-Z:DIM B1(22),B2(22),W(65),F1(12),F2(12)
240 DRAW "BM20,20;C2L3HL3HLBM+7,+3F2NGRNFENL2ENL4ENL3ER3ERBM-9,-1LDRDDBM50,20;L3GLGLG2BM+6,-3RF2NGRNFENL2ENL4R3FRF2BM-6,-4L3U3LDR":GET (11,17)-(30,24),B1:GET (41,17)-(60,24),B2
250 DRAW "BM10,100C3BD15BR4U4EUEUE5RERER3FRFGLGLGLGDGD2FDFRFRFNR2L17BE6P3,3":GET(10,100)-(31,115),W
260 F$="C1BD2E2R5F2G2L5H2BR3P1,1R7UED4HU":DRAW "BM100,100;XF$;BM150,100;A2;XF$;":GET(100,100)-(114,104),F1:GET(139,96)-(150,100),F2
270 SP$="C1RBR2H2L2G2F2R2E2BR7UH2LHL3HL4GL3GLG2D2F2R3FRFR4ER3ERE2UBR5U2HH3LHL3HL10GL3GLG3DGD4FD2F3RFR3FR10ER3ERE3U2EU2"
280 CLS:LOCATE 4,1:PRINT "PRESS THE NUMBER OF YOUR CHOICE:":PRINT:PRINT "  1)  JOYSTICK":PRINT:PRINT "  2)  KEYBOARD"
```

Continued

BIRD BRAIN *Continued*

IBM PC & IBM PCjr

```
290 A$=INKEY$:IF A$="" OR A$<"1" OR A$>"2" THEN 290 ELSE J=VAL(A$)
300 IF J=1 THEN PRINT:PRINT "PLACE YOUR JOYSTICK AT THE CENTER POSITION, AND PRESS [ENTER]." ELSE GOTO 320
310 A$=INKEY$:IF A$="" THEN 310 ELSE JC=STICK(0)
320 CLS:DRAW "BM130,60C3E3RE3RE4UEU2EU4R7ER3ER2E3U2ER5F2D4G2DG2DG2L2GL3GL3GL4GL6BE6P1,3BH3C3UH2LHL3HL12GL3GLG2D4FD2FD2L5HL4HL3HL3HL2H2UH2UH2U4E2R5FD2F3R2FR3FR7BG4P1,3BF6C3RF3RF3RF3BU4P3,3"
330 DRAW "BM-9,-26C1E2U5H2L5G2D5F2R5BL3U3ER4BH3P1,1BD4P3,1BM+4,-4C2R3EBU3NEG2LGHEUEUEUBH2BLD4GD2BL2U8BL2D8LU2HU2HUBG2BLF2DFDGNH4GHLHL"
340 DRAW "BM+30,+10C1H2U5E2R5F2D5G2L5BR3U3HL4BF2P3,1C1BU5P1,1BL3C2L3HBU3NHF2RFEHUHUHUBE2BRD4FD2BR2U8BR2D8RU2EU2EUBF2BRG2DGDFNE4FERER"
350 LOCATE 12,12:PRINT "BIRD-BRAIN":LOCATE 15,8:PRINT "ENTER YOUR CHOICE:":PRINT:PRINT SPC(5) "1) EASY":PRINT:PRINT SPC(5) "2) HARD":PRINT:PRINT SPC(5) "3) HARDER"
360 A$=INKEY$:IF A$="" OR A$<"1" OR A$>"3" THEN 360 ELSE LV=VAL(A$)
370 CLS:COLOR 3,0:LINE (0,143)-(319,167),3,BF:DRAW "BM0,136M+32,-8R8F4R4F2R2F2R2R24E4R12M+32,-12R16F2R18E8R12F10R22M+24,-8R16M+16,-4M+24,+12R8F8M+16,-6BM100,142P3,3"
380 G$="C1NR10RU2HBR3NED2BR2URNU2DBR2EUH":DRAW "BM72,144;XG$;BM+4,+6;XG$;BM+6,+6;XG$;BM+12,+1;XG$;BM+6,+6;XG$;BM+20,+4;XG$;BM40,140;XG$;BM54,142;XG$;BM45,146;XG$;BM145,126;XG$;BM155,126;XG$;BM194,128;XG$;BM204,128;XG$;BM260,140;XG$;"
390 DRAW "BM48,140C2E3U30HU8HU4HU2HU2HUH3UH3UH7R4F10DFDFDFD2FD2FDFDFD2FD2FD8E2UE2UE3UE4RE2RE2REF2GL2G3LG5DG2DGD4G2D3GD4GD16F3L14BE7P2,2"
400 DRAW "BM38,101C1ERE2UE2UEU3EU6H4UH3UH2LHLU2R4F2RFRF2RFRFR2F3RF2RF2RF3UHU3HU2HUHUH3LH2LH2L3HL13GL5GL2GLG3DG2DG2D3GD4GD8F2U5EUEUEUE2UE3UE2UERE2RERD3GD2GD2GD3GD4FD2FD2F4RFRBU4P1,1BH18P1,1"
410 DRAW "BM68,104C1ERERER2ER2E2U2R2FD3GD2GD5FE4UEU2EU5HUH3LHH2HL9GL3G5D4FUEUE2RERER5D2G2LG2DGD3BE5P1,1"
420 DRAW "BM24,62C2NR12GC3NR14DC2NR14DC3NR14FC2NR12FC3NR10FC2NR8R2FC3R2":BIRDS=3
430 B$="ERNFURNL3UNL3HNLU3LDR":IF BIRDS>0 THEN DRAW "C2BM29,61;XB$;" ELSE GOTO 890
440 IF BIRDS>1 THEN DRAW "C2BM68,87;XB$;"
450 IF BIRDS>2 THEN DRAW "C2BM84,94;XB$;"
460 AT=319:WP=0:PUT(WP,152),W:FP=150:GOSUB 700:IF FD=1 THEN PUT(FP,170),F1 ELSE PUT(FP,170),F2
470 IF J=1 THEN GOSUB 650 ELSE GOSUB 680
480 IF F=0 THEN 470 ELSE LINE(28,53)-(33,61),0,BF:APX=20:ABY=52:ABX=0:AY=0:LINE (0,197)-(AT-1,199),3,BF:PUT(APX,ABY),B1:GOSUB 710:GOSUB 540
490 IF J=1 THEN GOSUB 650 ELSE GOSUB 680
500 IF F=1 THEN GOSUB 710
510 GOSUB 540:GOSUB 590:AY=AY+(LV/12)+.1+AFISH:GOSUB 780:AT=AT-(.2+(LV/6)):IF AT<1 THEN 870 ELSE LINE (AT,197)-(AT,199),0:GOTO 490
520 GOSUB 590:AY=AY+(LV/12):GOSUB 780:GOTO 490
530 GOTO 530
540 PUT(APX,ABY),B1:IF FSH=1 THEN PUT(APX,ABY+6),F1
550 APX=APX+ABX:ABY=ABY+AY
560 IF APX>300 THEN APX=10 ELSE IF APX<10 THEN APX=300
570 IF ABY<5 THEN ABY=5:AY=.2 ELSE IF ABY>176 THEN RETURN 720
580 PUT(APX,ABY),B1:IF FSH=1 THEN PUT(APX,ABY+6),F1:RETURN ELSE RETURN
590 PUT(WP,152),W:WP=WP+LV*2:IF WP>298 THEN WP=0
600 PUT(WP,152),W
610 IF FSH=1 THEN RETURN ELSE IF FD<1 THEN 630 ELSE PUT(FP,170),F1:FP=FP-5:IF FP<0 THEN FP=300
620 PUT(FP,170),F1:RETURN
630 PUT(FP,170),F2:FP=FP+5:IF FP>300 THEN FP=0
640 PUT(FP,170),F2:RETURN
650 S=STICK(0):IF S<JC-20 THEN ABX=ABX-.25:IF ABX<-6 THEN ABX=-6:GOTO 670 ELSE GOTO 670
660 IF S>JC+20 THEN ABX=ABX+.25:IF ABX>6 THEN ABX=6
670 F=STRIG(0)*(-1):RETURN
680 F=0:A$=INKEY$:IF A$="" THEN RETURN ELSE IF A$="S" THEN ABX=ABX-.25:IF ABX<-6 THEN ABX=-6 ELSE ELSE IF A$="D" THEN ABX=ABX+.25:IF ABX>6 THEN ABX=6 ELSE ELSE IF A$=CHR$(32) THEN F=1 ELSE F=0
690 DEF SEG=0:POKE 1050,PEEK(1052):RETURN
700 FD=RND*2-1:IF FD=0 THEN 700 ELSE RETURN
710 PUT(APX,ABY),B1:PUT(APX,ABY),B2:SOUND 110,2:SOUND 157,2:PUT(APX,ABY),B2:PUT(APX,ABY),B1:AY=AY-(1/LV)-.5+AFISH:RETURN
720 AFISH=0:FOR SC=1 TO 6:DRAW "BM=APX;,=ABY;S=SC;XSP$;":SOUND 750-(SC*100),5:NEXT:DRAW "S4":BIRDS=BIRDS-1:IF BIRDS=0 THEN 890
730 LINE (0,168)-(319,199),0,BF:PUT (WP,152),W:IF BIRDS=1 THEN 760
740 DRAW "BM68,87;C0XB$;":APX=68:ABY=84:PUT(APX,ABY),B1:AY=-1:ABX=-1:FSH=0:FOR GONEST=1 TO 30:GOSUB 540:FOR TD=1 TO 50:NEXT:NEXT:AY=0:FOR GONEST=1 TO 17:GOSUB 540:FOR TD=1 TO 50:NEXT:NEXT
750 PUT(APX,ABY),B1:DRAW "C2BM29,61;XB$;":GOTO 460
760 DRAW "BM84,94;C0XB$;":APX=84:ABY=90:PUT(APX,ABY),B1:AY=-1:ABX=-1:FSH=0:FOR GONEST=1 TO 36:GOSUB 540:FOR TD=1 TO 50:NEXT:NEXT:AY=0:FOR GONEST=1 TO 27:GOSUB 540:FOR TD=1 TO 50:NEXT:NEXT
770 PUT(APX,ABY),B1:DRAW "C2BM29,61;XB$;":GOTO 460
780 IF ABY<162 OR ABY>172 OR APX<FP-2 OR APX>FP+4 THEN 810
790 IF FSH=1 THEN RETURN ELSE FSH=1:GOSUB 860:FOR SND=400 TO 2000 STEP 100:SOUND SND,1:NEXT:IF FD=1 THEN PUT(FP,170),F1 ELSE PUT(FP,170),F2
800 SOUND 440,2:PUT(APX,ABY+6),F1:RETURN
810 IF ABY<146 OR ABY>170 OR APX<WP OR APX>WP+22 THEN 830
820 AY=.2:ABX=3:SOUND 220,10:FOR Z=1 TO 20:GOSUB 540:NEXT:ABX=0:RETURN
830 IF ABY>54 OR ABY<50 OR APX>24 OR APX<16 THEN RETURN
840 PUT(APX,ABY),B1:DRAW "C2BM29,61;XB$;":IF FSH=1 THEN PUT(APX,ABY+6),F1:SCORE=SCORE+AT:FSH=0:AFISH=0:AT=319:GOSUB 700:FP=150:IF FD=1 THEN PUT(FP,170),F1 ELSE PUT(FP,170),F2
850 RETURN 470
860 AFISH=RND/2:IF ABS(AFISH-(1/LV)-.5)<(LV/12)+.1+AFISH THEN 860 ELSE RETURN
870 AY=1:SND=2000-(APY*10)
880 SOUND SND,1:GOSUB 540:SND=SND-10:GOTO 880
890 SCREEN 0:COLOR 7,0,0:SOUND 330,5:SOUND 220,10:SOUND 110,15:CLS:LOCATE 5,1:PRINT "ACCUMULATED TIME LEFT ON THE CLOCK FOR EACH FISH EATEN: ";:LOCATE 6,31:PRINT USING "########";SCORE
900 PRINT:PRINT "DIFFICULTY LEVEL:";:LOCATE 8,31:PRINT USING "X       #";LV:PRINT TAB(31);"--------";:PRINT:PRINT "TOTAL SCORE =";:LOCATE 11,31:PRINT USING "########";SCORE*LV
910 LOCATE 14,1:PRINT "WOULD YOU LIKE TO PLAY AGAIN (Y/N)?"
920 A$=INKEY$:IF A$<>"N" AND A$<>"Y" THEN 920 ELSE IF A$="N" THEN END ELSE SCORE=0:BIRDS=3:SCREEN 1:COLOR 0,0:GOTO 320
```

HCM

PROGRAM LISTING

BIRD BRAIN

TI-99/4A

```
100 REM ***************
110 REM * BIRD-BRAIN *
120 REM ***************
130 REM BY CRAIG BLAZAKIS
140 REM HOME COMPUTER MAGAZINE
150 REM VERSION 4.5.1
160 REM TI EXTENDED BASIC
170 REM
180 REM TITLE SCREEN
190 CALL MAGNIFY(3)
200 CALL CHAR(100,"05155554F2F9F7F383020201010080807404844AF2F4F9FE1C0C0404080810 10E0")
210 CALL CHAR(104,"0000000000000000010303030100000000000000000000080C0C0408000000 00")
220 CALL CHAR(108,"000007182040402018070000000000000000FF0020400000000000000E05C4221100F")
```

Continued

```
230 CALL CHAR(112,"F00F0000000000F0
    0738424284000000
    0001800000000018
240 RT=1
250 CALL CLEAR :: CALL SCREEN(16)
260 CALL SPRITE(#1,100,7,30,100,#2,100,
    7,30,125,#3,104,2,32,100,#4,104,2,2
    8,125)
270 CALL SPRITE(#5,108,11,50,105,#6,112
    ,11,50,121)
280 RANDOMIZE
290 DISPLAY AT(10,8):"  BIRD BRAIN"
300 REM
310 X=5 :: FOR D=1 TO 20 :: X=-X :: CAL
    L MOTION(#3,X,0,#4,0,300,0,-4,0)::
320 CALL SOUND(10,100,0) :: FOR T=1 TO
    10 :: NEXT T
330 NEXT D
340 CALL MOTION(#3,0,0,#4,0,0)::
    CALL CHAR(#3,"30")
350 CALL CLEAR :: DISPLAY AT(12,5):"PRE
    SS 1) FOR JOYSTICK":"2) FOR KEYBOARD
    "
    ACCEPT AT(13,26)VALIDATE(
    "12")SIZE(1) :: CALL JOY(1,X,Y)
360 CALL CLEAR :: DISPLAY AT(15,5):"1 =
    PLAY AT (10,8)":"2 = BIRD BRAIN"
370 DISPLAY AT(17,5):"2 = HARDER"
380 DISPLAY AT(19,5):"3 = DIFFICULTY LE
    VEL"
390 DISPLAY AT(10,3):"DIFFICULTY LEVEL"
    ACCEPT AT(10,26)VALIDATE(
    "123")SIZE(1)
400 IF RT=1 THEN CALL SOUND(70,700,0):
    :CALL SOUND(70,#4,32,#3,123):: CALL
    DELSPRITE(ALL) :: GOTO 410
410 FOR D=1 TO 150 :: NEXT D
420 RT=0
430 IF DLEVEL=2 THEN Q=3 :: R=1.8 :: TO
    ME=128 :: TM=4
440 IF DLEVEL=3 THEN Q=2 :: R=2 :: TOME
    =96
450 IF DLEVEL=1 THEN Q=4 :: R=1 :: TOME
    =160
460 CALL DELSPRITE(ALL) :: CALL CLEAR
470 DISPLAY AT(10,6):"PLEASE STAND BY"
480 REM CHAR DEFINITIONS
490 DATA 1C0800C1C0800C0E0C0C0E0C0C0707
    0E0C0707070E0F0F0F0F0F0F0F0F0F0F0F0
    30E0F0F0F0F0F0F0F0F0F0F0F0F0F0F0F0F
500 DATA 0103E3E1F1F1F1F1F1F1F1F1F1F1F
    1F1F1F1F1F1F1F1F1F1F1F1F1F1F1F1F1F1
    F1F1F1F1F1F1F1F1F1F1F1F1F1F1F1F1F1F
510 DATA 7C7C7C7C7C7C7C7C7C7C7C7C7C7C7
    C7C7C7C7C7C7C7C7C7C7C7C7C7C7C7C7C7C
520 DATA 0000000000000000000000000000F
    0F0F0F0F0F0F0F0F0F0F0F0F0F0F0F0F0F0
    F0F0F0F0F0F0F0F0F0F0F0F0F0F0F0F0F0F
530 DATA FFFFFFFFFFFFFFFFFFFFFFFFFFFFF
    FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
    FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
540 DATA 0000000000000000000000000000F
    FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
    FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
550 DATA 0000000000000000000000000000F
    FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
    FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
560 DATA 0000000000000000000000000000F
    FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
    FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
570 CALL CHAR(39,"FFFFFFFFFFFFFFFF")
580 CALL CHAR(98,"0000000000000000")
590 CALL CHAR(99,"0000000000000000")
600 CALL CHAR(140,"0000000000000000")
610 CALL CHAR(108,"0000000000000000
    3838463838000000
    6C6C6C6C6C000000
620 CALL CHAR(112,"0000000000000000
    0000000000000000
    0000000000000000
630 CALL CHAR(100,"0000000000000000
    0000000000000000
    0000000000000000
640 CALL CHAR(60,"0000000000000000
    0000000000000000
    0000000000000000
650 CALL CHAR(136,"0000000000000000
    0000000000000000
    0000000000000000
660 CALL CHAR(104,"0000000000000000
    0000000000000000
    0000000000000000
670 CALL CHAR(116,"0000000000000000
    0000000000000000
    0000000000000000
680 CALL CHAR(120,"0000000000000000
    0000000000000000
    0000000000000000
690 CALL CHAR(128,"0000000000000000
    0000000000000000
    0000000000000000
700 CALL CHAR(132,"0000000000000000
    0000000000000000
    0000000000000000
710 REM SET UP SCREEN
720 CALL CLEAR
730 CALL SCREEN(15)
740 CALL COLOR(2,7,1,3,7,1,6,5,5,7,3,1,
    8,3,1,9,3,1,1,11,1,14,2,5)
750 FOR X=40 TO 55 :: READ A$ :: CALL C
    HAR(X,A$) :: NEXT X :: READ A$ :: CA
    LL CHAR(X,A$)
760 FOR X=80 TO 91 :: READ A$ :: CALL C
    HAR(X,A$) :: NEXT X :: READ A$ :: CA
    LL CHAR(X,A$)
770 FOR X=92 TO 97 :: READ A$ :: CALL C
    HAR(X,A$) :: NEXT X :: READ A$ :: CA
    LL CHAR(X,A$)
780 FOR X=33 TO 38 :: READ A$ :: CALL C
    HAR(X,A$) :: NEXT X
790 DATA 12,6,7,13,8,13,7,14,7,15,7,15
    ,8,16,7,8,13,8,17,7,17,8,18,7,18,8,
    19,7,8,19,5,8,16,9,4,9,11,4,10,8,5,
    10,6,10,8,11,4,10,13,8,13,9,14,8,14
    ,9,14,10,15,10
800 DATA 37,38,33,34,35,36,53,54,98,99
    ,98,99,99,99,33,33,34,35,38,33,34,3
    5,36
810 DATA 37,40,36,33,38,33,33,99,99,99
    ,99,99,99,99,99,99,99,99,99,99,99,9
    9,99,99,99,99,99,99,99,99,99,99,99,
    99,99,99,99,99,99,99,99,99,99,99,99
820 DATA 37,40,U,V :: CALL HCHAR(12,5,
    X,32) :: IF X=1 THEN 830 ELSE 840
830 CALL HCHAR(19,C,C,CH) :: C=C+1 :: I
    F C=CH+1 THEN C=1 :: CH=CH+1 :: IF C
    H=32 THEN CH=1
840 NEXT X
850 CALL HCHAR(U,V,X,92) :: NEXT X
860 CALL HCHAR(U,V,X,97) :: NEXT X
870 CALL HCHAR(20,1,39,64)
880 FOR X=1 TO 32 :: READ CH :: CALL H
    CHAR(19,C,CH) :: C=C+1 :: NEXT X
890 CALL MAGNIFY(3)
900 CALL SPRITE(#20,120,7,55,30)
910 CALL SPRITE(#13,112,16,12,100,#14,1
    12,29,175,#15,112,16,18,200)
920 GOSUB 1750
930 CALL SPRITE(#10,124,2,161,1,0,SPEED
    ,1,#11,136,5,153,1,0,DLEVEL+2+1)
940 CALL SPRITE(#1,136,5,153,1,0,DLEVEL
    ,1,#2,116,2,55,30)
950 CALL SPRITE(#3,116,2,55,30)
960 CALL SPRITE(#4,116,2,100,70,#5,116,
    94,57)
970 CALL HCHAR(22,1,72,96)
980 CALL HCHAR(24,1,140,32)
990 REM PROGRAM LOGIC
1000 RANDOMIZE
1010 SPN=3 :: SCORE=0
1020 GOTO 1240
1030 CALL KEY(0,K1,S1)
1040 CALL HCHAR(7,1,0,32) :: CALL PATTE
     RN(#3,C2) :: CALL SOUND(10,-8,0)
1050 IF TIME=TOME THEN 1100 ELSE TIME=T
     IME+1 :: IF TIME/TM=INT(TIME/TM) T
     HEN CALL HCHAR(24,72,96) :: GOTO 10
     30
1060 IF K1=18 OR K1=32 THEN 1070 ELSE I
     F K1=83 OR K1=115 THEN 1080 ELSE IF
     K1=68 OR K1=100 THEN 1090 ELSE 1030
1070 IF S=3 THEN S=0 :: GOTO 1100 ELSE
     S=3
1080 IF S=3 THEN S=0 :: GOTO 1100 ELSE
     S=3
1090 IF S=3 THEN S=0 :: GOTO 1100 ELSE
     S=3
1100 CALL SOUND(-50,-1,0) :: CALL SOUND
     (-50,-2,0) :: CALL SOUND(-50,-3,0)
     :: CALL SOUND(-50,-4,0)
1110 IF X<60 AND Y>25 AND Y<35 THEN 112
     0 ELSE 1130
1120 CALL COINC(#3,#10,15,0,#12,#13,0)
     :: IF C=0 THEN 1130 ELSE CALL DELSP
     RITE(#1,#2,#3,#10,#11,#12,#13,#14,#
     15) :: CALL SOUND(100,500,0) :: CAL
     L SOUND(100,-5,0) :: CALL SOUND(100
     ,-8,0) :: CALL SOUND(100,-1,0) :: C
     ALL SOUND(100,-5,0) :: CALL SOUND(1
     00,-8,0) :: CALL SOUND(100,-1,0) ::
     C1=1 :: C2=1 :: GOTO 1240
1130 CALL MOTION(#3,0,0,#4,0,0,#10,0,0,
     #11,0,0,#12,0,0,#13,0,0,#14,0,0,#15
     ,0,0)
1140 CALL MOTION(#3,0,0)
1150 CALL MOTION(#3,0,0)
1160 CALL MOTION(#3,0,0)
1170 CALL MOTION(#3,0,0)
1180 FOR X=1 TO 5 :: CALL SOUND(-500,-6
     ,2*X) :: NEXT X
1190 CALL DELSPRITE(ALL) :: GOTO 20
1200 NEXT D
1210 FOR D=1 TO 20 :: NEXT D
1220 CALL PRITE(#3)
1230 FOR D=1 TO 300 :: NEXT D :: SPN=SP
     N-1 :: IF SPN=0 THEN 1750 ELSE CALL
     DELSPRITE(#3) :: GOTO 1240
1240 CALL SPRITE(#3,55,30,0,0,#4,55,30,
     0,0,#5,55,30,0,0,#6,55,30,0,0,#7,55
     ,30,0,0,#8,55,30,0,0,#9,55,30,0,0)
1250 IF SH=1 THEN CALL SOUND(100,500,0,
     500,0,-8,0,500,0,-1,0) :: SH=0 :: G
     OTO 1260 ELSE IF SH=3 THEN 1260 ELS
     E CALL SOUND(200,750,0,140,0,-8,0,1
     40,0,-1,0) :: SH=0
1260 IF RI=1 THEN 1270 ELSE RI=RI+1 :: I
     F RI=2 THEN CALL SPRITE(#10,124,2,1
     61,1,0,SPEED,1,#11,136,5,153,1,0,DL
     EVEL+2+1) :: GOTO 1280 ELSE IF RI=3
     THEN CALL SPRITE(#10,124,2,161,1,0,
     SPEED,1,#11,136,5,153,1,0,DLEVEL+2+
     1) :: GOTO 1280 ELSE IF RI=4 THEN C
     ALL SPRITE(#10,124,2,161,1,0,SPEED,
     1,#11,136,5,153,1,0,DLEVEL+2+1) :: G
     OTO 1280 ELSE IF RI=5 THEN CALL SPR
     ITE(#10,124,2,161,1,0,SPEED,1,#11,1
     36,5,153,1,0,DLEVEL+2+1) :: GOTO 12
     80 ELSE IF RI=6 THEN CALL SPRITE(#1
     0,124,2,161,1,0,SPEED,1,#11,136,5,1
     53,1,0,DLEVEL+2+1) :: GOTO 1280 ELS
     E CALL SPRITE(#10,124,2,161,1,0,SPE
     ED,1,#11,136,5,153,1,0,DLEVEL+2+1)
1270 CALL KEY(1,KY,S)
1280 IF S=0 THEN 1290 ELSE IF KY=68 OR
     KY=100 THEN 1300 ELSE IF KY=83 OR K
     Y=115 THEN 1310 ELSE 1320
1290 CALL JOYST(1,A,B) :: IF A=0 AND B=
     0 THEN 1320 ELSE CALL JOYST(1,A,B)
     :: S=1 :: IF A=4 THEN 1300 ELSE IF
     A=-4 THEN 1310 ELSE 1320
1300 CALL MOTION(#1,0,4,#2,0,4,#3,0,4,#
     4,0,4,#5,0,4,#6,0,4,#7,0,4,#8,0,4,#
     9,0,4) :: GOTO 1320
1310 CALL MOTION(#1,0,-4,#2,0,-4,#3,0,-
     4,#4,0,-4,#5,0,-4,#6,0,-4,#7,0,-4,#
     8,0,-4,#9,0,-4)
1320 CALL COINC(#3,#10,15,0,#12,#13,0)
     :: IF C=0 THEN 1330 ELSE CALL DELSP
     RITE(#1,#2,#3,#10,#11,#12,#13,#14,#
     15) :: CALL SOUND(100,500,0) :: CAL
     L SOUND(100,-5,0) :: CALL SOUND(100
     ,-8,0) :: CALL SOUND(100,-1,0) :: C
     ALL SOUND(100,-5,0) :: CALL SOUND(1
     00,-8,0) :: CALL SOUND(100,-1,0) ::
     C1=1 :: C2=1 :: GOTO 1240
1330 REM BIRD
```

Continued

BIRD BRAIN *Continued*

TI-99/4A

```
1340 FISH=0 :: CALL POSITION(#3,116,2,AP,BP):: CALL SPRITE(#SP,
     ::: CALL DELSPRITE(#SP)
     LL CALL PRINT(24,1,"TIME=170
1350 C: CALL HCHAR(24,1,140,32):: TIME=170
     :CALL PLAYSONG(33)
1360 :CALL PLAYSONG("T TERN(#3,104)::  CALL SOUND(5
     0,CALL PLAY"M0 TION(#3,-10,0)
1370 CFOR L=X=PAT 1000
1380 CALL LINK("TO TERN(#3,100)::  CALL POSITIO
1390 N(#3,CALL PLAY"T TERN(#3,100)::  IF THEN 1430
1400 FCALL PLAY"T TERN(#3,104)::  CALL SOUND(5
1410 0,CALL PLAY"M0 TION(#3,-10,0)
1420 NEXT P
1430 CALL PLAY"T TERN(#3,100)::  CALL MOTION(
     #3,CALL LINK"-10)  TO 1000  CALL POSITION(#3
1440 F,APPLY"X B P)::  IF BP<33 THEN 1460
1450 NEXT X
1460 CALL MOTION(#3,5,0)
1470 FOR X=1 TO 100 STEP 2::  CALL POSITION(#
     3,APPLY"X B P)::  IF AP>52 THEN 1240
1480 NEXT X
1490 REM END OF GAME SCORE
1500 CLEAR ::  CALL CHAR(42,"0000FFFFFFFFFFFF")
     ELSE CALL PRINT"CHEAT(ALL)::  BOOM"
1510 CALL SPRITE("42,1,0,0,0,0)
1520 DISPLAY AT(9,1):"LEVEL 10 ":"TOTAL SCORE
     DMESSAGE$="CLOCK FOR SCORE":: DELAY=1000
     ::FOR D=1 TO 19 STEP 2::  CALL HCHAR(D,1,
1530 D"; ";D+1,1,"  ",32):: NEXT D
1540 DISPLAY AT(9,1):"LEVEL 11 ":"TOTAL SCORE
1550 DISPLAY AT(11,1):"COLOR LEVEL ":"LEVEL
1560 IF SCORE=1600 THEN 1570 ELSE 1580
1570 HIGH SCORE
1580 NEXT
1590 REM
1600 DISPLAY AT(14,22):"HIGH SCORE
1610 DISPLAY AT(14,1):"PLAY AGAIN (Y/N)?
1620 "&SCREEN COLOR 16
1630 IF Y" THEN 350 ELSE CALL CLEAR
1640 REM OUTPUT
1650 CALL MOTION(#3,0,0)
1660 FOR X=1 TO 5 :: CALL LINK
     L SOUND(500,-7,0):: NEXT X :: CALL
     NCALL SOUND(400,110,0)
1670 CALL SOUND(400,-3,AP,BP)
1680 CALL SPRITE(#17,-5,0,-5,0)
     CALL PRINT"SCORE 1730
1690 CALL PATTERN(#3,60)
1700 CALL MOTION(#3,-5,0)
1710 FOR X=1 TO 10 STEP 2::  CALL POSITION(#
     3,AP,BP)::  IF AP<10 THEN 1730
     NCALL SOUND(-400,-3,AP,BP)
     EALL SPRITE(#17)
1740 SPN=6 THEN 1500 ELSE
1750 ERANDOMIZE
1760 IF CALL CHAR(124,"0000000000000000")
     CALL CHAR(96,"0000000000000000")
     S"0000000000000000")
     RETURN
```

HCM

DIVISION TUTOR

APPLE // Family

```
100 REM  ********** TUTOR **********
110 REM  ***** DIVISION *****
120 REM  ***** BY LISTSONBEE *****
130 REM  ***** THE COMPUTER MAGAZINE *****
140 REM  HOME VERSION 4.5.1
150 REM  VAPPLE FAMILY APPLESOFT
160 REM
170 REM
180 REM  PROGRAM INITIALIZATION
190 GOSUB 28000: REM  INITIALIZATION
200 GOSUB 45200: REM  TITLE SCREEN
210 IF F$ = "" THEN GOTO 230
220 IF F$ = "2" THEN 240
230 F$ = "1": HOME : GOTO 270
240 PRINT CHR$ (7);: GOTO 220
250 REM
260 REM  INITIALIZATION
270 REM
280 ESCAPE$ = CHR$ (27): REM
290 BACKSLASH$ = CHR$ (92): REM
300 REM
310 REM  TITLE BOX COORDINATES
320 BX(1) = 8: BX(2) = 7: BX(3) = 18: BX(4
) = 8: BX(5) = 18: BX(6) = 8: BX(7) = 18: BX(8) = 8
330 BX$ = CHR$ (42)
340 UNDERSCORE$ = CHR$ (95): REM
350 REM
360 REM  DETERMINES POSITION OF
370 REM  OPERATIONS
380 DIM X(2)
390 DIM Y(2)
400 FOR I = 1 TO 57: READ X: POKE 768 + I, X:
    NEXT
410 RETURN "DIVIDEND","MULTIPLIER"
420 DATA 173,48,168,192,201,1,208,237,160,1
    ,173,136,169,208,192,252,1,169,136,133
    ,252,173,169,5,133,253,96
430 DATA 173,48,168,192,201,208,237,160,1,1
    73,136,169,208,192,252,1,169,136,133
    ,252,173,169,5,133,253,96
440 REM
450 REM
460 HOME : VTAB 1: HTAB 1: GOSUB 590
470 GOSUB 600: DIVISION TUTOR
480 GOSUB 610
490 GOSUB 590
500 GOTO 610
510 HTAB 9: VTAB 11
520 PRINT "FLASHCARD DIVISION
530 PRINT
540 PRINT "LONG DIVISION
550 RETURN
560 REM
570 REM
580 REM
590 PRINT "
600 PRINT "
610 PRINT "
620 PRINT "
630 PRINT "
640 PRINT "
650 PRINT "
660 PRINT "
670 PRINT "
680 PRINT "
690 PRINT "
700 PRINT "
710 PRINT "
720 PRINT "
730 PRINT "
740 REM
750 REM
760 PRINT "
770 REM
780 REM
790 REM
800 REM
810 REM
820 REM
830 REM
840 REM
850 REM
860 REM
870 REM
880 REM
890 REM
900 REM
910 REM
920 REM
930 REM
940 REM
950 REM
960 REM
970 REM
980 REM
990 REM
1000 REM
1010 REM
```

Continued

DIVISION TUTOR *Continued*

PROGRAM LISTING

```
10200 VTAB 9: HTAB 10: PRINT "
10300 VTAB 9: HTAB 11 + PX: PRINT B / A
10400 FOR D I = 1 TO 1500: NEXT
10500 GOTO 820
10600 RETURN
10700 REM NEW FLASHCARD PROBLEM
10800 VTAB 8: FOR IT = 1 TO 7
10900 PRINT " ": NEXT
11000 VTAB 11: HTAB 10: PRINT " )"
11100 VTAB 10: HTAB 10: PRINT UN$UN$UN$UN$UN
$UN
11200 VTAB 11: HTAB 9: IF A > 9 THEN HTA
B 8
11300 PRINT A
11400 VTAB 11: HTAB 11: PRINT B
11500 RETURN
11600 REM ENTER FLASHCARD ANSWER
11700 VTAB 9: HTAB 10: PRINT "
11800 VTAB 9: " HTAB 11 + PX
11900 FA$ = B 32 40 ESC$ THEN RETURN
12000 GOSUB 32 40 ESC$ "0" THEN RETURN
12100 IF IN$ = "0" AND IN$ < > "9" THEN
12200 IF IN$ = INT (FA$) AND IN$ = FA$ THEN
      IF LEN (STR$ (B / A)) = LEN (STR$
      (FA$)) THEN
      HEN GOTO 1200
12300 IF (IN$ = CHR$ (13)) OR (LEN (FA$
      ) = LEN (STR$ (A))) THEN FA$ =
      VAL (FA$): RETURN
12400 PRINT CHR$ (7);: GOTO 1200
12500 RETURN
12600 REM LONG DIVISION
12700 GOSUB 1540
12800 GOSUB 3020
12900 GOSUB 1650
13000 IF IN$ = ESC$ THEN RETURN
13100 GOSUB 1740
13200 A = INT (( RND (1) * CA) + 2: B = B
      - A + 1
      NT (( RND (1) * A) + 1): PB$ = "
      TP (1): DW = 1: PB$ = "
13300 TP (2) = 1: DW = 1: PB$ = "
      : GOSUB 2010: TP(2) = TP(2) * 10
      + DW: DW = DW + 1: GOTO 1330
13400 FOR IT = 1 TO LEN ( STR$ (B)) - TP
      (2): TO = TO * 10: NEXT
13500 FOR IT = 1 TO LEN ( STR$ (B)) - LE
      N ( STR$ (TP(2))): PB$ = PB$ +
      MID$ ( STR$ (A), THEN 1380
13600 PB$ = PB$ + MID$ ( STR$ (B)), TP(1)
      ), TP (2))
13700 NEXT
13800 PB$ = MID$ ( STR$ (B)), TP(1)), TP
      (2))
      : GOSUB 2080: PB$ = VAL (PB$)
13900 GOSUB 2080
14000 IF IN$ = ESC$ THEN RETURN
14100 IF IN$ = "0" THEN TP(1) = TP(1) +
      1: GOTO 1480
14200 GOSUB 2080
14300 IF IN$ = ESC$ THEN RETURN
14400 GOSUB 2080
14500 IF IN$ = ESC$ THEN RETURN
14600 TP(1) = TP(1) + TP(2): IF TP(1) =
      LEN ( STR$ (B)) THEN DW = DW +
      1: GOTO 1500
14700 IF LEN ( STR$ (TP(1))) < > LEN (
      STR$ (B)) THEN
14800 IF TP(1) = 2470 GOSUB 2700
14900 IF TP(1) = LEN ( STR$ (B)) THEN
      13900
15000 GOSUB 2700
15100 GOSUB 2920
15200 IF IN$ < > ESC$ THEN 1320
15300 RETURN
15400 REM LONG DIV CHOICE SCREEN
15500 HOME
15600 VTAB 3: HTAB 3
15700 PRINT " WELCOME TO THE LONG DIVISION
      TUTOR.:"
15800 VTAB 5: HTAB 3
15900 PRINT "CHOOSE A LEVEL OF DIFFICULTY
      :"
16000 VTAB 9: HTAB 15
16100 PRINT "1) BEGINNING"
16200 VTAB 12: HTAB 15
16300 PRINT "2) ADVANCED"
16400 RETURN CHOOSE DIFFICULTY LEVEL
16500 GOSUB 3400
16600 IF IN$ = "1" THEN DF = 1: GOTO 1710
16700 IF IN$ = "2" THEN DF = 2: GOTO 1710
16800 IF IN$ = ESC$ THEN RETURN
16900 PRINT CHR$ (7);: GOTO 1660
17000 IF DF = 1 THEN CA = 8: CB = 8
17100 URN DF = 2 THEN CA = 98: CB = 98
17200 RETURN
17300 REM LONG DIVISION BACKGROUND
17400 HOME : VTAB 1: GOSUB 1920
17500 FOR IT = 2 TO 22: VTAB IT: HTAB 1:
      PRINT "*";: HTAB 40: PRINT "*";:
      NE
      XT
17600 VTAB 22: GOSUB 1920
17700 FOR IT = 2 TO 21: VTAB IT: HTAB 2:
      PRINT "*";: HTAB 39: PRINT "*";:
      NEXT
17800 VTAB 1: HTAB 38: INVERSE : PRINT U
      N$;: NORMAL : PRINT UN$;
17900 VTAB IT: HTAB 2: INVERSE : PRINT U
      N$;: NORMAL : PRINT UN$;
18000 FOR JI = 1 TO 19: INVERSE : PRINT "
      ";: NORMAL : PRINT UN$;: NEXT
18100 HTAB JI: INVERSE : PRINT UN$;: NOR
      MAL : PRINT UN$;
18200 VTAB IT + 1: HTAB 2: INVERSE : PRI
      NT UN$;: NORMAL : PRINT UN$;
18300 FOR JI = 1 TO 19: INVERSE : PRINT "
      ";: NORMAL : PRINT UN$;: NEXT
18400 NEXT : NORMAL
18500 FOR IT = 1 TO 6: VTAB IT: GOSUB 195
      0: NEXT
18600 VTAB 7: GOSUB 1960: VTAB 22: GOSUB
      19600
18700 GOSUB 1970
18800 VTAB IT: FOR IT = 1 TO 5: HTAB 15:
      PRINT UN$;: NEXT
18900 GOSUB 1990
19000 RETURN
19100 REM MISCELLANEOUS LONG DIV SCREEN
19200 HTAB 1: PRINT "**********";: RETURN
19300 GOSUB 3020
19400 RETURN
19500 HTAB 10: PRINT "!"
      !": RETURN
19600 HTAB 4: PRINT "=========";: RETURN
19700 FOR IT = 8 TO 21: VTAB IT
19800 HTAB 4: PRINT "!";: NEXT : RETURN
19900 FOR IT = 2 TO 6: VTAB IT: HTAB 11:
      PRINT "|";: NEXT
20000 VTAB 11: PRINT "--|--";
20100 REM LONG DIV PROBLEM
20200 FOR IT = 8 TO 21
20300 VTAB IT: HTAB 5: PRINT "
20400 VTAB IT: HTAB 8 LEN ( STR$ (A))
20500 PRINT B
20600 VTAB 9: HTAB 8: FOR IT = 0 TO LEN
      ( STR$ (B)): PRINT UN$;: NEXT
20700 RETURN
20800 REM LONG DIV ENTRY
20900 PX (1) = PX (2) A
21000 RT = GOSUB 1990: GOSUB 2360
21100 CT = 0
21200 VTAB 8: HTAB 7 + TP(1) + TP(2)
21300 GOSUB 2400: IF IN$ = ESC$ THEN RET
      URN
21400 PRINT IN$;: IF IN$ < > "9" AND IN$
      < > "0" THEN
21500 IF IN$ = "0" AND CT = 0 THEN
21600 PRINT CHR$ (7);: FOR DI = 1 TO 200
21700 IF GOSUB 3510: GOTO 2200
21800 GOSUB 3470: CT = CT + 1
21900 IF VTAB 3: HTAB 7 + TP(1) + TP(2):
      PRI
      NT
22000 VTAB 8: HTAB 7 + TP(1) + TP(2)
22100 RETURN
22200 REM LONG DIV ENTRY 2
22300 PX (2) = PX (1) GOSUB 1990: GOSUB
      2360
22400 PZ = 0
22500 IF IN$ = "0" GOSUB 3070
22600 IF IN$ = ESC$ THEN RETURN
22700 VTAB 9: HTAB 8 + TP(1) + TP(2)
      PF (2) = 0 THEN
      OR IT = 1 TO TP(2): PZ = PZ * 10:
      NEXT
22800 IF IN$ = "0" THEN 2200
22900 RETURN
23000 REM LONG DIV ENTRY 3
23100 RT = PX (1): GOSUB 1990: GOSUB 2360
23200 PB = PB - PX (2): PB$ = STR$ (PB)
23300 PY$ = "": C = 0: PZ = PB
23400 PD = 0: GOSUB 3070
23500 RETURN
23600 REM MOMENTARY PROBLEM
23700 VTAB 10: HTAB 22: PRINT "
23800 IF RT = 4 THEN RETURN
23900 VTAB 10: HTAB 23
24000 PRINT PX (1);
24100 ON RT GOTO 24200,24300,24400
24200 PRINT "+";: GOTO 24500
24300 PRINT "-";: GOTO 24500
24400 PRINT "X";
24500 PRINT PX (2);: INPUT "= ?";
24600 RETURN
24700 REM BRING DOWN
24800 RT = 4: GOSUB 1990: GOSUB 2360
24900 BD$ = MID$ ( STR$ (B), TP (1), 1)
25000 GOSUB 2570
25100 PB$ = PB$ + BD$: PB = PB * 10 + VA
      L (BD$)
25200 VLEN = LEN (PB$)
25300 RT = 4: GOSUB 1990
25400 PB$ = STR$ (PB)
25500 RETURN
25600 REM BRING DOWN GRAPHICS
25700 IF RT = 1 THEN 0: R2 = 100: R3 = 4
25800 FOR R = 1 TO 6
25900 FOR VERT = 0 TO 10: HTAB 8 + TP(1)
      : VTAB VERT + 1: INVERSE : PRINT
      UN$;: NORMAL : PRINT UN$;
26000 NEXT
26100 RETURN
26200 VTAB 10: HTAB 8 + TP(1) + TP(2)
26300 PRINT B
26400 RETURN
26500 FOR JI = 1 TO DW * 3 + 7: INVERSE
      : PRINT UN$;: NORMAL : PRINT UN$;
      : NEXT
26600 VTAB 8 + TP(1): HTAB 8: PRINT "
```

Continued

DIVISION TUTOR *Continued*

APPLE // Family

```
2670 NORMAL : VTAB JI: HTAB 8 + TP(1): NORMAL : PRINT BD$: GOSUB 3630: GOSUB 2890
2680 VTAB JI: HTAB 8 + TP(1): PRINT " ": NEXT
2690 RETURN
2700 REM REMAINDER
2710 RT = 5: GOSUB 1990: GOSUB 2360
2720 GOSUB 2760
2730 HTAB 10 + TP(1): VTAB 8
2740 PRINT "R "PB
2750 RETURN
2760 REM ARROWS TO REMAINDER
2770 DX = 10:R1 = 1:R2 = 200:R3 = 5
2780 FOR IT = 9 TO 10: VTAB DW * 3 + 7: HTAB IT + TP(1)
2790 INVERSE : PRINT ">": GOSUB 3630: GOSUB 2890
2800 VTAB DW * 3 + 7: HTAB IT + TP(1)
2810 NORMAL : PRINT ">": GOSUB 3630: GOSUB 2890: NEXT
2820 FOR IT = DW * 3 + 6 TO 9 STEP - 1
2830 HTAB 11 + TP(1): VTAB IT
2840 INVERSE : PRINT "^": GOSUB 3630: GOSUB 2890
2850 HTAB 11 + TP(1): VTAB IT
2860 NORMAL : PRINT "^": GOSUB 3630: GOSUB 2890
2870 NEXT
2880 RETURN
2890 REM ADJUSTABLE DELAY LOOP
2900 FOR DI = 1 TO DX: NEXT
2910 RETURN
2920 REM MORE LONGDIV?
2930 VTAB 13: HTAB 22: PRINT "FOR MORE, PRESS"
2940 VTAB 15: HTAB 25: FLASH : PRINT "<RETURN>"
2950 FOR DI = 1 TO 1200: NEXT
2960 VTAB 15: HTAB 25: INVERSE : PRINT "<RETURN>": NORMAL
2970 VTAB 17: HTAB 28: GOSUB 3240
2980 IF IN$ < > CHR$ (13) AND IN$ < > ESC$ THEN 2970
2990 VTAB 13: HTAB 22: PRINT " "
3000 VTAB 15: HTAB 22: PRINT " "
3010 RETURN
3020 REM ESCAPE MESSAGE
3030 VTAB 23: HTAB 10: INVERSE
3040 PRINT "PRESS 'ESC' TO LEAVE"
3050 NORMAL
3060 RETURN
3070 REM LONGDIV NUMBER ENTRY
3080 VTAB PD + 3 * DW: HTAB TP(1) + TP(2) + 8 - LEN ( STR$ (PZ)): PRINT " "
3090 PY$ = "": VTAB PD + 3 * DW: HTAB TP(1) + TP(2) + 8 - LEN ( STR$ (PZ))
3100 GOSUB 3240: IF IN$ = ESC$ THEN RETURN
3110 IF IN$ = CHR$ (13) THEN 3170
3120 IF ASC (IN$) > 31 THEN PRINT IN$;: GOTO 3140
3130 PRINT " ";
3140 IF IN$ > = "0" AND IN$ < = "9" THEN 3160
3150 PRINT CHR$ (7) CHR$ (8);: GOTO 3100
3160 PY$ = PY$ + IN$: IF LEN (PY$) < 3 THEN 3100
3170 GOSUB 3220
3180 IF VAL (PY$) = PZ THEN GOSUB 3510: GOTO 3210
3190 GOSUB 3470: IF CT = 2 THEN 3210
3200 CT = CT + 1: GOTO 3080
3210 PY$ = STR$ (PZ): GOSUB 3220: RETURN
3220 VTAB PD + 3 * DW: HTAB TP(1) + TP(2) + 8 - LEN ( STR$ (PZ)):PY$ = STR$ ( VAL (PY$)): PRINT "" CHR$ (8) CHR$ (8) CHR$ (8) CHR$ (8) CHR$ (8)PY$
3230 RETURN
3240 REM GET CHARACTER WITH PROMPT
3250 POKE - 16368,0:DX = 4
3260 NORMAL : PRINT "?" CHR$ (8);
3270 FOR DI = 1 TO DX: GOSUB 3360
3280 IF CX = 1 THEN 3340
3290 NEXT
3300 NORMAL : PRINT UN$ CHR$ (8);
3310 FOR DI = 1 TO DX: GOSUB 3360
3320 IF CX = 1 THEN 3340
3330 NEXT : GOTO 3260
3340 NORMAL : PRINT " " CHR$ (8);
3350 RETURN
3360 CX = 0:X = PEEK ( - 16384) > 127
3370 ON X + 1 GOTO 3380,3390
3380 RETURN
3390 POKE - 16368,0:IN$ = CHR$ ( PEEK ( - 16384))
3400 CX = 1
3410 RETURN
3420 REM GET CHARACTER
3430 X = PEEK ( - 16384) > 127
3440 ON X + 1 GOTO 3430,3450
3450 POKE - 16368,0:IN$ = CHR$ ( PEEK ( - 16384))
3460 RETURN
3470 REM ERROR SOUND
3480 POKE S3,0
3490 CALL SD
3500 RETURN
3510 REM CORRECT SOUND
3520 R1 = 4 + INT (10 * RND (1))
3530 R2 = 200
3540 FOR ZI = 1 TO R1
3550 R3 = R2 + INT (40 * RND (1))
3560 POKE S1,255 - R3
3570 POKE S2,R3
3580 POKE S3,1
3590 CALL SD
3600 R3 = R3 - 15
3610 NEXT
3620 RETURN
3630 REM ASCENDING OR DESCENDING SOUND
3640 IF R1 < > 0 THEN R2 = R2 - R3: GOTO 3660
3650 R2 = R2 + R3
3660 POKE S1,255 - R2
3670 POKE S2,R2
3680 POKE S3,1
3690 CALL SD
3700 RETURN
```

HCM

PROGRAM LISTING

DIVISION TUTOR

COMMODORE 64

```
100 REM ******************
110 REM * DIVISION TUTOR *
120 REM ******************
130 REM BY STEVEN LISONBEE
140 REM AND THE HCM STAFF
150 REM HOME COMPUTER MAGAZINE
160 REM VERSION 4.5.1
170 REM C-64 BASIC
180 REM
190 REM
200 PRINT"{SHIFT CLR}{CTRL GRN}";:POKE 53280,0:POKE 53281,0:POKE 54296,15
210 PRINT"{13CRSRDOWN}                    ** DIVISION TUTOR **"
220 PRINT"{9CRSRDOWN}                PRESS {CTRL RVSON}RETURN{CTRL RVSOFF} TO BEGIN"
230 POKE 198,0
240 GET A$:IF A$<>CHR$(13) THEN 240
250 PRINT"{SHIFT CLR}{CRSRDOWN}        ** MAIN MENU **"
260 PRINT"{5CRSRDOWN}{CTRL CYN}PRESS F1 TO SELECT FLASHCARD DIVISION"
270 PRINT"{2CRSRDOWN}{CTRL GRN}F3 TO SELECT LONG DIVISION"
280 PRINT"{2CRSRDOWN}{CMDR BLU}F5 TO EXIT PROGRAM{CTRL GRN}"
290 POKE 198,0
300 GET A$:IF A$="" THEN 300
310 IF A$="{F1}" THEN 410:REM FLASHCARD
320 IF A$="{F3}" THEN 350:REM LONG DIVISION
330 IF A$="{F5}" THEN 2600:REM EXIT
340 GOTO 300
350 PRINT"{SHIFT CLR}{2CRSRDOWN}PRESS F1 FOR BEGINNERS LEVEL"
360 PRINT"{2CRSRDOWN}PRESS F3 FOR ADVANCED LEVEL"
370 POKE 198,0
380 GET A$:IF A$<>"{F1}" AND A$<>"{F3}" THEN 380
390 IF A$="{F1}" THEN CA=9:CB=400:GOTO 790
400 IF A$="{F3}" THEN CA=30:CB=20:GOTO 790
410 PRINT"{SHIFT CLR}{CTRL CYN}":R=23:C=8:GOSUB 2140:PRINT"PRESS F5 TO RETURN TO MENU"
420 PRINT"{HOME}{2CRSRDOWN}{39CMDR T}";
430 PRINT"{HOME}{3CRSRDOWN}  RIGHT:  WRONG:       SCORE:"
440 PRINT"{39CMDR T}"
450 CR$="{HOME}{3CRSRDOWN}{8CRSRRIGHT}":CW$=CR$+"{12CRSRRIGHT}":CS$=CW$+"{14CRSRRIGHT}"
460 RT=0:WR=0:MRK=0
470 A=INT(11*RND(1)+2):B=INT(11*RND(1)+2):B=A*B:CT=0
480 IF MRK=0 OR(WR=0 AND RT=0)THEN SC=0:GOTO 500
490 SC=INT(RT/(RT+WR)*100+.5):SC$=MID$(STR$(SC),2)
500 IF SC=0 THEN SC$="0"
510 PRINT"{CTRL CYN}";CR$;RT;CW$;WR;CS$;SC$;"%"
520 R=9:C=16:GOSUB 2140:PRINT"     "
530 A$=STR$(A):B$=STR$(B):C$=STR$(B/A):D=LEN(B$)-LEN(C$)
```

Continued

DIVISION TUTOR *Continued*


```
540 R=10:C=16:GOSUB 2140:PRINT"
     "
550 C=18:GOSUB 2140:FOR I=1 TO LEN(B$):
    PRINT"{CMDR @}";:NEXT I
560 R=11:C=15:GOSUB 2140:PRINT"
     "
570 R=11:C=18-LEN(A$):GOSUB 2140:PRINT
    A$;")";MID$(B$,2)
580 R=9:C=19+D:GOSUB 2140
590 LN=2:GOSUB 2170
600 IF EX THEN 250
610 MRK=1:DIV=Q
620 IF DIV<>B/A THEN 740
630 POKE 54277,0:POKE 54278,128:POKE 54
    276,65:POKE 54275,8:POKE 54274,0
640 POKE 54272,0
650 FL=0:FOR PITCH=20 TO 40
660 POKE 54273,PITCH
670 R=17:C=12:GOSUB 2140
680 IF FL=0 OR FL=8 OR FL=16 THEN PRINT
    "VERY GOOD WORK"
690 IF FL=4 OR FL=12 OR FL=20 THEN PRIN
    T"               "
700 FL=FL+1:IF FL=21 THEN FL=0
710 NEXT PI:RT=RT+1:FOR I=54272 TO 5429
    5:POKE I,0:NEXT I
720 FOR TD=1 TO 1000:NEXT TD
730 GOTO 470
740 R=17:C=10:GOSUB 2360
750 IF CT<2 THEN 480
760 R=17:C=8:R$=MID$(STR$(B/A),2)
770 GOSUB 2490
780 WR=WR+1:GOTO 470
790 PRINT"{SHIFT CLR}{CMDR WHT}";:MRK=0
800 FOR BR=1 TO 12
810 PRINT"{CTRL RVSON}{SHIFT O}{SHIFT P}{SHIFT O}{SHIFT P}{SHIFT O}{SHIFT P}{SHIFT O}{SHIFT P}{SHIFT O}{SHIFT P}{SHIFT O}{SHIFT P}{SHIFT O}{SHIFT P}{SHIFT O}{SHIFT P}{SHIFT O}{SHIFT P}{SHIFT O}{SHIFT P}{SHIFT O}{SHIFT P}{SHIFT O}{SHIFT P}{SHIFT O}{SHIFT P}{SHIFT O}{SHIFT P}{SHIFT O}{SHIFT P}{SHIFT O}{SHIFT P}{SHIFT O}{SHIFT P}{SHIFT O}{SHIFT P}{SHIFT O}{SHIFT P}{SHIFT O}{SHIFT P}{SHIFT O}{SHIFT P}{SHIFT O}"
820 PRINT"{CTRL RVSON}{SHIFT P}{SHIFT O}{SHIFT P}{SHIFT O}{SHIFT P}{SHIFT O}{SHIFT P}{SHIFT O}{SHIFT P}{SHIFT O}{SHIFT P}{SHIFT O}{SHIFT P}{SHIFT O}{SHIFT P}{SHIFT O}{SHIFT P}{SHIFT O}{SHIFT P}{SHIFT O}{SHIFT P}{SHIFT O}{SHIFT P}{SHIFT O}{SHIFT P}{SHIFT O}{SHIFT P}{SHIFT O}{SHIFT P}{SHIFT O}{SHIFT P}{SHIFT O}{SHIFT P}{SHIFT O}{SHIFT P}{SHIFT O}{SHIFT P}{SHIFT O}{SHIFT P}{SHIFT O}{SHIFT P}{SHIFT O}{SHIFT P}"
830 NEXT BR
840 PRINT"{CTRL RVSON}{SHIFT O}{SHIFT P}{SHIFT O}{SHIFT P}{SHIFT O}{SHIFT P}{SHIFT O}{SHIFT P}{SHIFT O}{SHIFT P}{SHIFT O}{SHIFT P}{SHIFT O}{SHIFT P}{SHIFT O}{SHIFT P}{SHIFT O}{SHIFT P}{SHIFT O}{SHIFT P}{SHIFT O}{SHIFT P}{SHIFT O}{SHIFT P}{SHIFT O}{SHIFT P}{SHIFT O}{SHIFT P}{SHIFT O}{SHIFT P}{SHIFT O}{SHIFT P}{SHIFT O}{SHIFT P}{SHIFT O}{SHIFT P}{SHIFT O}{SHIFT P}{SHIFT O}{SHIFT P}{SHIFT O}{SHIFT P}{SHIFT O}";
850 PRINT"{HOME}{CTRL CYN}"
860 PRINT"{8 CRSR RIGHT}{CTRL RVSON}{SHIFT O}{20 CMDR Y}{SHIFT P}"
870 FOR OB=1 TO 5
880 PRINT"{8 CRSR RIGHT}{CTRL RVSON}{CMDR H}                        {CMDR N}"
890 NEXT OB
900 PRINT"{HOME}{2 CRSR DOWN}{11 CRSR RIGHT}{CTRL RVSON}DIVIDE"
910 PRINT"{11 CRSR RIGHT}{CTRL RVSON}MULT
    IPLY"
920 PRINT"{11 CRSR RIGHT}{CTRL RVSON}SUBT
    RACT"
930 PRINT"{11 CRSR RIGHT}{CTRL RVSON}BRIN
    G DOWN"
940 PRINT"{11 CRSR RIGHT}{CTRL RVSON}REMA
    INDER"
950 PRINT"{CTRL GRN}"
960 PRINT"{5 CRSR RIGHT}{SHIFT O}{26 CMDR Y}{SHIFT P}"
970 FOR BB=1 TO 14
980 PRINT"{5 CRSR RIGHT}{CMDR H}                          {CMDR N}"
990 NEXT BB
1000 PRINT"{5 CRSR RIGHT}{SHIFT L}{26 CMDR P}{SHIFT @}"
1010 A=INT(CA*RND(1)+2):B=INT(CB*RND(1)+
     2):B=B*A+INT(10*RND(1))
1020 A$=MID$(STR$(A),2):B$=MID$(STR$(B),
     2)
1030 R=10:C=20:GOSUB 2140
1040 FOR SG=1 TO LEN(B$)+1:PRINT"{CMDR @}
     ";:NEXT SG
1050 R=11:C=20-LEN(A$):GOSUB 2140
1060 PRINT A$;")";B$
1070 ROW=10:COL=21+LEN(B$)-LEN(MID$(STR$
     (INT(B/A)),2))
1080 FOR PROB=1 TO LEN(B$)
1090 GOSUB 2090
1100 ARO=2:GOSUB 2110:SD=F
1110 IF MRK=0 THEN PROB=COL-20:SD=INT(VA
     L(MID$(B$,1,PROB)))
1120 CT=0
1130 R=9:C=6:GOSUB 2140
1140 PRINT"{CTRL RVSON}"MID$(STR$(SD),2)
     "/"A$" = ?"
1150 R=9:C=COL:GOSUB 2140
1160 LN=1:GOSUB 2170:D=Q
1170 IF EX THEN 250:REM ESCAPE
1180 IF D=INT(SD/A) THEN 1250
1190 R=22:C=9:GOSUB 2360
1200 IF CT<3 THEN 1150
1210 D=INT(SD/A):R$=MID$(STR$(D),2)
1220 R=22:C=6:GOSUB 2490
1230 R=9:C=COL:GOSUB 2140
1240 PRINT R$:R$="":REM PUT CORRECT NUMB
     ER ON SCREEN
1250 COL=COL+1:ROW=ROW+2:D$=MID$(STR$(D)
     ,2):DA$=MID$(STR$(D*A),2)
1260 IF (D=0)*(PROB=LEN(B$)) THEN 1800:R
     EM REMAINDER TIME
1270 ARO=ARO+1:GOSUB 2090:GOSUB 2110
1280 CT=0
1290 R=9:C=6:GOSUB 2140
1300 PRINT"{CTRL RVSON}"D$"*"A$" = ?{CTR
     L RVSOFF}"
1310 R=ROW:C=COL-LEN(DA$):GOSUB 2140
1320 LN=LEN(DA$):GOSUB 2170:M=Q
1330 IF EX THEN 250
1340 IF M=D*A THEN 1410
1350 R=22:C=9:GOSUB 2360
1360 IF CT<3 THEN 1310
1370 M=D*A:R$=MID$(STR$(M),2)
1380 R=22:C=6:GOSUB 2490
1390 R=ROW:C=COL-LEN(DA$):GOSUB 2140
1400 PRINT DA$
1410 R=ROW+1:C=COL-LEN(DA$):GOSUB 2140
1420 FOR P=1 TO LEN(DA$):PRINT"{SHIFT C}
     ";:NEXT P
1430 ARO=ARO+1:GOSUB 2090:GOSUB 2110
1440 ROW=ROW+2:SD=F:IF MRK=0 THEN SD=VAL
     (MID$(B$,1,PROB))
1450 CT=0
1460 SP=LEN(MID$(STR$(SD-M),2))
1470 R=9:C=6:GOSUB 2140
1480 PRINT"{CTRL RVSON}"MID$(STR$(SD),2)
     "-"DA$" = ?{CTRL RVSOFF}"
1490 R=ROW:C=COL-SP:GOSUB 2140
1500 LN=SP:GOSUB 2170:S=Q
1510 IF EX THEN 250
1520 IF SD-M=S THEN R$=MID$(STR$(S),2):G
     OTO 1570
1530 R=22:C=9:GOSUB 2360
1540 IF CT<3 THEN 1490
1550 S=SD-M:R$=MID$(STR$(S),2)
1560 R=22:C=6:GOSUB 2490
1570 R=ROW:C=COL-SP:GOSUB 2140
1580 PRINT R$
1590 R=9:C=6:GOSUB 2140
1600 PRINT"                  "
1610 F=S:IF PROB=LEN(B$) THEN 1800
1620 ARO=ARO+1:GOSUB 2090:GOSUB 2110
1630 CT=0
1640 R=12:C=COL:GOSUB 2140
1650 PRINT"{CTRL RVSON}↑{CTRL RVSOFF}"
1660 R=ROW:C=COL:GOSUB 2140
1670 LN=1:GOSUB 2170:BD=Q
1680 IF EX THEN 250
1690 R=12:C=COL:GOSUB 2140
1700 PRINT" ":REM ERASE ARROW
1710 IF VAL(MID$(B$,PROB+1,1))=BD THEN 1
     780
1720 R=22:C=9:GOSUB 2360
1730 IF CT<2 THEN 1640
1740 BD=VAL(MID$(B$,PROB+1,1)):R$=MID$(S
     TR$(BD),2)
1750 R=22:C=6:GOSUB 2490
1760 R=ROW:C=COL:GOSUB 2140
1770 PRINT R$
1780 MRK=1:ROW=ROW-1:F=S*10+BD
1790 NEXT PROB
1800 MRK=0
1810 R=9:C=6:GOSUB 2140:PRINT"
     "
1820 GOSUB 2090:ARO=6:GOSUB 2110
1830 R=9:C=COL+1:GOSUB 2140
1840 PRINT"R"
1850 CT=0
1860 IF CH>1 THEN 1890
1870 R=ROW:C=COL+2:GOSUB 2140:PRINT"↑":R
     =ROW-5:C=COL+2::GOSUB 2140:PRINT"↑"
1880 R=10:C=COL+2:GOSUB 2140:PRINT"↑"
1890 R=9:C=COL+2:GOSUB 2140
1900 LN=1:GOSUB 2170:R=Q
1910 IF EX THEN STOP
1920 IF R=F THEN 2000
1930 R=22:C=9:GOSUB 2360
1940 IF CT<2 THEN 1890
1950 IF D>0 THEN R$=MID$(STR$(S),2)
1960 IF D=0 THEN R$=MID$(STR$(F),2)
1970 R=22:C=6:GOSUB 2490
1980 R=9:C=COL+2:GOSUB 2140
1990 PRINT R$
2000 R=22:C=6:GOSUB 2140
2010 PRINT"PRESS ANY KEY TO CONTINUE"
2020 POKE 198,0
2030 GET A$:IF A$="" THEN 2030
2040 GOSUB 2090
```

Continued

PROGRAM LISTING

DIVISION TUTOR *Continued*

COMMODORE 64

```
2050 PRINT"{HOME}{8CRSRDOWN}"
2060 FOR ER=1 TO 14:PRINT"{6CRSRRIGHT}":NEXT ER
2070 GOTO 1010
2080 PRINT"{HOME}{CTRL GRN}";:END
2090 PRINT"{HOME}{2CRSRDOWN}{10CRSRRIGHT}{CTRL RVSON}{CTRL CYN}  {SHIFT CRSRLEFT}{CRSRDOWN} {SHIFT CRSRLEFT}{CRSRDOWN} {SHIFT CRSRLEFT}{CRSRDOWN} {SHIFT CRSRLEFT}{CRSRDOWN} {CTRL GRN}"
2100 RETURN
2110 R=ARO:C=10:GOSUB 2140
2120 PRINT"{CTRL RVSON}{CTRL CYN}{SHIFT Q}{CTRL GRN}"
2130 RETURN
2140 POKE 781,R:POKE 782,C:POKE 783,0
2150 SYS 65520
2160 RETURN
2170 EX=0:TM=0:TB=0:CU=2:CU$="{CMDR +}  ":DL$=CHR$(20):Q$=""
2180 NU$=DL$+"0123456789"
2190 POKE 198,0:PRINT"{CTRL GRN}     {4SHIFT CRSRLEFT}";
2200 GET N$:IF N$<>"" THEN 2230
2210 IF TI>TM THEN PRINT MID$(CU$,CU,1);"{SHIFT CRSRLEFT}";:CU=3-CU:TM=TI+15
2220 GOTO 2200
2230 IF N$=CHR$(13) THEN 2350
2240 IF N$="{F5}" THEN EX=1:RETURN
2250 FOR CK=1 TO 11:IF MID$(NU$,CK,1)=N$ THEN 2270
2260 NEXT CK:GOTO 2200
2270 IF TB=0 AND N$=DL$ THEN 2200
2280 IF TB=LN AND N$>DL$ THEN 2200
2290 IF N$>DL$ THEN PRINT "{CTRL GRN}"N$"{CTRL GRN}";:GOTO 2310
2300 PRINT" {2SHIFT CRSRLEFT}";:REM DELETES
2310 TB=TB+1+2*(N$=DL$)
2320 Q$=Q$+N$
2330 IF N$=DL$ THEN Q$=LEFT$(Q$,LEN(Q$)-2)
2340 GOTO 2200
2350 PRINT"  ":Q=VAL(Q$):RETURN
2360 GOSUB 2140
2370 POKE 54277,0:POKE 54278,128:POKE 54276,65:POKE 54275,8:POKE 54274,0
2380 POKE 54272,70
2390 PRINT"THAT IS NOT CORRECT"
2400 FOR PITCH=30 TO 10 STEP-1
2410 POKE 54273,PITCH
2420 FOR TD=1 TO 15:NEXT TD
2430 NEXT PITCH
2440 FOR SR=54272 TO 54295:POKE SR,0:NEXT SR
2450 FOR TD=1 TO 500:NEXT TD
2460 GOSUB 2140
2470 PRINT"                    "
2480 CT=CT+1:RETURN
2490 GOSUB 2140
2500 PRINT"THE CORRECT ANSWER IS  ";
2510 FOR AN=1 TO 4
2520 PRINT"   {3SHIFT CRSRLEFT}";
2530 FOR TD=1 TO 200:NEXT TD
2540 PRINT "{CTRL RVSON}"R$"{CTRL RVSOFF}";LEFT$("{3SHIFT CRSRLEFT}",LEN(R$));
2550 FOR TD=1 TO 200:NEXT TD
2560 NEXT AN
2570 FOR TD=1 TO 1000:NEXT TD
2580 PRINT:PRINT"{SHIFT CRSRUP}{6CRSRRIGHT}"
2590 RETURN
2600 PRINT"{SHIFT CLR}{CTRL GRN}";:POKE 54296,0:REM POKE 808,237
```

HCM

DIVISION TUTOR

IBM PC & IBM PCjr

```
100 ' ****************
110 ' * DIVISION TUTOR *
120 ' ****************
130 ' BY STEVEN LISONBEE
140 ' AND THE HCM STAFF
150 ' HOME COMPUTER MAGAZINE
160 ' VERSION 4.5.1
170 ' IBM PCjr CARTRIDGE BASIC
      FROM DOS 2.1
180 ' IBM PC BASICA WITH
190 ' COLOR/GRAPHICS MONITOR ADAPTER
200 ' AND COLOR MONITOR
210 ' INITIALIZE PROGRAM
220 '
230 KEY OFF:SCREEN 0:WIDTH 40:COLOR 3,0,0:CLS:RANDOMIZE TIMER:PLAY "MF"
240 DIM B$(1)
250 FOR Z=1 TO 20:B$(0)=B$(0)+CHR$(193)+CHR$(194):B$(1)=B$(1)+CHR$(194)+CHR$(193):NEXT
260 ' TITLE SCREEN
270 GOSUB 860:GOSUB 870:LOCATE 12,7:PRINT "D I V I S I O N - T U T O R":LOCATE 18,8:PRINT "PRESS [ENTER] TO CONTINUE"
280 A$=INKEY$:IF A$<>CHR$(13) THEN 280 ELSE GOSUB 870
290 ' OPTION MENU
300 RT=0:WR=RT:LOCATE 7,13:PRINT "DIVISION-TUTOR":LOCATE 10,8:PRINT "1) FLASHCARD DIVISION":LOCATE 12,8:PRINT "2) LONG DIVISION":LOCATE 14,8:PRINT "3) EXIT PROGRAM":LOCATE 18,10:PRINT "YOUR CHOICE?";
310 A$=INKEY$:IF A$<"1" OR A$>"3" THEN 310 ELSE CH=VAL(A$):IF CH=3 THEN CLS:SCREEN 0:PRINT "BYE BYE.. SEE YOU IN CLASS.":END ELSE IF CH=2 THEN GOTO 450
320 ' FLASHCARD ROUTINE
330 COLOR 3,3,3:CLS:FOR Z=2 TO 6:COLOR 7,0,3:LOCATE Z+1,6:PRINT SPC(30):COLOR 8,7,3:LOCATE Z,5:PRINT SPC(30);:NEXT:FOR Z=9 TO 18:COLOR 7,0,3:LOCATE Z+1,16:PRINT SPC(10);:COLOR 0,7,3:LOCATE Z,15:PRINT SPC(10);:NEXT
340 LOCATE 3,7:PRINT "RIGHT:";TAB(25);"WRONG:";:LOCATE 5,15:PRINT "SCORE:";
350 CT=0:A=INT(RND*11)+2:ANSWER=INT(RND*11)+2:B=A*ANSWER:A$=STR$(A):B$=STR$(B):C$=STR$(ANSWER)
360 LOCATE 12,15:PRINT SPC(10):LOCATE 13,15:PRINT SPC(10):LOCATE 14,15:PRINT SPC(10)
370 LOCATE 13,19:PRINT STRING$(LEN(B$)+2,95):LOCATE 14,16:PRINT USING "##)###";A,B
380 CN=LEN(C$)-1:R=12:C=24-LEN(C$):GOSUB 890 : IF ANS=0 THEN COLOR 3,3,3:CLS:COLOR 3,0,3:GOSUB 870:GOTO 300
390 IF ANS=0 THEN 270
400 IF ANS=ANSWER THEN GOSUB 930:RT=RT+1:SCORE=INT(RT/(RT+WR)*100):LOCATE 3,12:PRINT RT:LOCATE 5,21:PRINT SCORE:GOTO 350
410 CT=CT+1:SOUND 220,5:SOUND 110,8:IF CT=1 THEN LOCATE 12,20:PRINT "  ";:GOTO 380
420 SOUND 220,5:SOUND 110,10:LOCATE 22,9:COLOR 0,3,3:PRINT "THE RIGHT ANSWER IS:":COLOR 0,7,3:LOCATE 12,23-LEN(C$):PRINT C$:FOR TD=1 TO 3000:NEXT:LOCATE 22,9:COLOR 0,3,3:PRINT SPC(20);:COLOR 0,7,3:WR=WR+1:SCORE=INT(RT/(RT+WR)*100)
430 LOCATE 3,31:PRINT WR:LOCATE 5,21:PRINT SCORE:GOTO 350
440 ' LONG DIVISION ROUTINE
450 GOSUB 870:LOCATE 5,8:PRINT "CHOOSE A LEVEL:":LOCATE 8,10:PRINT "1) BEGINNER":LOCATE 10,10:PRINT "2) ADVANCED"
460 A$=INKEY$:IF A$<"1" OR A$>"2" THEN 460 ELSE IF A$="1" THEN CHA=9:CHB=400 ELSE CHA=30:CHB=20
470 COLOR 0,4,4:CLS:FOR Z=1 TO 12:PRINT B$(0);:PRINT B$(1);:NEXT:COLOR 9,4:LOCATE 1,11:PRINT STRING$(18,220):COLOR 0,15:FOR Z=2 TO 6:LOCATE Z,12:PRINT SPC(16):NEXT
480 LOCATE 8,10:COLOR 6,4:PRINT STRING$(20,220):COLOR 6,0:FOR Z=9 TO 22:LOCATE Z,10:PRINT CHR$(221);SPC(18);CHR$(222):NEXT:LOCATE 23,10:COLOR 6,4:PRINT STRING$(20,223)
490 RESTORE 500:COLOR 0,15,4:FOR Z=2 TO 6:READ A$:LOCATE Z,13:PRINT A$:NEXT
500 DATA DIVIDE,MULTIPLY,SUBTRACT,BRING DOWN,REMAINDER
510 A=INT(RND*CHA)+2:B=INT(RND*CHB)+2:B=B*A+INT(RND*10):A$=STR$(A):B$=STR$(B)
520 COLOR 7,0:LOCATE 12,17:PRINT STRING$(LEN(B$)+2,95):LOCATE 13,16-LEN(A$):PRINT A;")";B
530 ROW=14:COL=19+LEN(B$)-LEN(STR$(INT(B/A))):ARO=2
540 FOR PROB=1 TO LEN(B$):ARO=5:GOSUB 970:ARO=6:GOSUB 970:GOSUB 970:ARO=2:CT=0:IF MARK=0 THEN PROB=COL-17
550 IF MARK=0 AND VAL(LEFT$(B$,PROB))<A THEN PROB=PROB+1
560 GOSUB 960:IF MARK=0 THEN SD=INT(VAL(LEFT$(B$,PROB))) ELSE SD=F
570 GOSUB 820:CN=1:R=11:C=COL:GOSUB 890:D=ANS:IF MARK=0 AND D=0 THEN COLOR 3,3,3:CLS:COLOR 3,0,3:GOSUB 870:GOTO 300
580 IF INT(SD/A)<>D THEN GOSUB 740:IF CT=3 THEN GOSUB 760 ELSE GOTO 570
590 GOSUB 970:ARO=3:COL=COL+1:D$=STR$(D):DA=D*A:DA$=STR$(DA):IF D=0 AND PROB=LEN(B$) THEN ROW=ROW-1:GOTO 680 ELSE GOSUB 960:CT=0:GOSUB 830
600 R=ROW:C=COL-LEN(DA$)+1:CN=LEN(DA$)-1:GOSUB 890:M=ANS:IF M<>D*A THEN GOSUB 740:IF CT=3 THEN GOSUB 760 ELSE LOCATE R,C:PRINT SPC(CN):GOTO 600
```

Continued

DIVISION TUTOR *Continued*

IBM PC & IBM PCjr

```
610 LOCATE R+1,C:PRINT STRING$(LEN(DA$),95):GOTO SUB 970:R=VAL(LEFT$(B$,PRO
    ARK=0 THEN SD=VAL(LEFT$(B$,PRO-1)):GOTO SUB 960:CN=LEN(STR$(SD-M))-1:CT=0
620 GOSUB 960:CN=LEN(STR$(SD-M))-1:CT=0
    :GOSUB 840
630 R=R+ROW<>S THEN C=COL-CN:GOSUB 890:S=ANS:IF S<>CT THEN S
    D-M:SUB 760 ELSE GOSUB 740:PRINT SPC
    GOTO 630
640 LOCATE LEN(B$)+1,11:PRINT SPC(18);:F=S:IF P
    ROW:IF F=1 THEN 670
650 GOSUB 970:ARO+1,1):FOR Z=14 TO 960:BDW-1,
    MID$(B$,PRO,Z,COL,BD$):INTERCEPT Z,CTR):C+Z-1:K
    LOCATE 11,COL:PRINT SPC(18)
    R$(32)+"?":SOUND (Z/2)*200,2:FOR D=1 TO COL-1:PR
    INTARK=1:ROW=ROW+1:F=S*10+VAL(BD$)
660 NEXT
670 MARK=0
680 ' PLAY THE REMAINDER
690 GOSUB 970:ARO=6:GOSUB 960:LOCATE 11
    ,OW TO SUB+0$:12:PRINTEP Z-1:LOCATE PRINT
    NT R$:LOCATE Z-1,C"
    )*200:NEXT:INSUB 930
700 COLOR 1,0,4:TOCATE 24,7:PRINT "PRESS
    A$=INKEY$:IF A$<>CHR$(13) THEN 720
    ELSE 0:FOR Z=9 TO 22:LOCATE 24
    Z,1:COLOR 0,4:SPRINT SPC(38);
    ' INCORRECT ANSWER ROUTINE
710 CT=CT+1:LOCATE 24,10:COLOR 0,4:PRINT
    CHAT IS NOT CORRECT
    5:SOUND 220,10:SOUND 200:NEXT 7,0:RE
    PC(20);:COLOR 0:RETURN
    ' GIVE THE CORRECT ANSWER
720 COLOR 0,4:LOCATE 24,12:PRINT "THE A
    NSWER IS";:FOR Z=1 TO 24:PRINT SPC(38);:NEXT:LOCATE 24
    ,1:COLOR 7,0:RETURN
730 ' INCORRECT ANSWER ROUTINE
740 CT=CT+1:LOCATE 24,10:COLOR 0,4:PRINT
    "THE A
    ANSWER IS";:FOR Z=1 TO 24:PRINT SPC(38);:NEXT:LOCATE 24
    ,1:COLOR 7,0:RETURN
750 ' GIVE THE CORRECT ANSWER
760 COLOR 0,4:LOCATE 24,12:PRINT "THE A
    NSWER IS";:FOR Z=1 TO 24:PRINT SPC(38);:NEXT:LOCATE 24
    ,1:COLOR 7,0:RETURN
770 D=INT((SD/A)):LOCATE 11,COL:PRINT RIG
    HT$((D))-1):RETURN
780 M=D*A:LOCATE ROW,COL-LEN((DA$)):PRINT
    DA$:RETURN
790 S=SD-M:RETURN
800 RETURN
810 ' DISPLAY ROUTINES FOR SMALL EQUATI
    ON
820 LOCATE 9,11:COLOR 7,0:PRINT SPC(18)
    ;:L?CATE 9,14:PRINT SD;CHR$(246);A;
    =:LOCATE 9,11:COLOR 7,0:PRINT SPC(18)
    ;:LOCATE 9,15:PRINT D;"X";A;"=";?:R
    ETURN
840 LOCATE 9,11:COLOR 7,0:PRINT SPC(18)
    ;:LOCATE 9,13:PRINT SD;"-";DA;"=?"
    :RETURN
850 ' DISPLAY ROUTINES FOR THE MENU AND
    BORDER
860 COLOR 3,3,3:CLS:RETURN
870 COLOR 0,3,3:FOR Z=4 TO 22:LOCATE Z,
    6:PRINT SPC(30);:NEXT Z:COLOR 3,0,3
    :RETURN
880 ' INSTRUCTIONS
890 ANS=0:POKE 1050,PEEK(1050)
    2):FOR Z=1 TO 9:NEXT:LOCATE R,C+Z-1:K
    $=INKEY$:IF K$="" THEN 900 ELSE IF
    K$<0 OR K$>9 THEN SOUND 110,1:G
    OTO 900:ELSE ANS=ANS+K$
    NEXT:SOUND 1,1:RETURN
910 ' MUSIC
920 ' PLAY CORRECT ANSWERS
930 ' DEFINE 12 NOTE SCALE
    ' RETURN
940 ' DISPLAY THE ARROW FOR
950 ' THE LONG DIVISION ROUTINE
960 COLOR 0,7:LOCATE ARO,12:PRINT CHR$(
    26);:COLOR 7,0:RETURN
970 COLOR 0,7:LOCATE ARO,12:PRINT CHR$(
    32);:COLOR 7,0:RETURN
```

HCM

DIVISION TUTOR

TI-99/4A

```
100 REM ********** DIVISION TUTOR **********
110 REM ********** LISTED BY MAGAZINE
120 REM ********** VERSION 4.5.1
130 REM BY SOME EXTENDED BASIC
140 REM HOME COMPUTER MAGAZINE
150 REM VERSION 4.5.1
160 REM EXTENDED BASIC
170 REM
180 REM *INITIALIZE*
190 CALL CLEAR::CALL SCREEN(15)::::::CALL
    CALL COLOR(13,7,1)::::RANDOMIZE::::OFF
    ARFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF")
200 REM ********DEFINE CHARACTERS FOR TITLE
    SCREEN
210 CALL CHAR(109,"00FC222121212121212121FC")::
    CALL CHAR(110,"00212121212121212121C000")::
    CALL CHAR(111,"0038101010101010103800")::
    CALL CHAR(112,"0010101010101010101000")::
    CALL CHAR(113,"0022141414141414140800")::
    CALL CHAR(116,"0020201010101010101C00")::
    CALL CHAR(117,"0020201010101010101C00")::
    CALL CHAR(118,"00FC414141414141417F00")::
    CALL CHAR(119,"00FC2C2A2A2A2A2A2A2900")::
    CALL CHAR(120,"0092928A8A8A8A8A868000")
    CALL CHAR(121,"")
270 REM INITIALIZE SCREEN
280 CALL HCHAR(3,4,128,27)::::CALL HCHAR
    (21,4,128,27)::::CALL HCHAR(4,30,128,18)::::
    19)::::::::::::::::::::::::::::::::::::::::
290 DISPLAY AT(10,7)ERASE ALL:"DIVISION TUTOR":::
    :"by Home Computer Magazine":::::"Version 4.5":::
    :"Extended BASIC"
300 CALL HCHAR(4,1,96,768)::::CALL HCHAR(
    95,1,96,768)::::CALL HCHAR(12,9,32,15)::::
    (11,9,32,15)::::::::::::::::::::::::::::::::
    8)
360 CALL CHAR(104,"FF8080808080808080FF01
    0101010101010101FF")::::CALL HCHAR(11
    FC0C0C0C0C0C0C0C0C0FFFF0303030303030FFFF")
370 CALL CHAR(64,"C0C0C0C0C0C0C0C0C0C0C0C0
    FFFFAR(116,"FFFF",117,"C0C0C0C0C0C0C0C0
    CHR$(D))-1):RETURN
    CALL HCHAR(89,Y$)::::CALLL
380 CALL CHAR(120,"FF00000000FF00FFFF88
    8888888888888888888888888888888888888888
    1313232,"FFFFFFFFFFFFFFFF")::"DO YOU WAN
    T TO DIVIDE?":"1. LONG DIVISION":"2. DIVISION
    WITH REMAINDERS":"3. END PROGRAM"
390 DISPLAY AT(10,5)SIZE(22)::"1. LONG DIVISION":
    "2. DIVISION WITH REMAINDERS":"3. END PROGRAM"
400 DISPLAY AT(16,5)SIZE(22)::"3. END PROGRAM"
410 DISPLAY AT(23,3)SIZE(18)::"ENTER YOUR CHOICE"
    ::::ACCEPT AT(23,21)VALIDATE(DIGIT):CH
420 CALL HCHAR(10,7,32,22)::::CALL HCHAR(11,7,32,22)
    ::::CALL HCHAR(12,7,32,22)::::CALL HCHAR(13,7,32,22)
    ::::CALL HCHAR(16,7,32,28)
430 ON CH GOSUB 450,480,3,440
440 REM FLASH CARD SCREEN
450 DISPLAY AT(7,9)SIZE(13)::"WELCOME TO FLASH CARDS"
    ::::::"ASK ME ADDITION":"SUBTRACTION":"MULTIPLICATION":"OR DIVISION"
    ::::"ENTER YOUR CHOICE"
460 FOR LP=1 TO 300::::NEXT LP::::CALL CLEAR
    ::::::::::::GOTO 530
470 REM SCREEN
480 DISPLAY AT(5,10)SIZE(10)::"WELCOME TO LONG
    DIVISION":::::"ENTER YOUR CHOICE"
490 DISPLAY AT(12,5)SIZE(22)::"1. BEGIN":"2. ADVANCED
    DIVISION":"3. PLAY CARDS"
500 DISPLAY AT(23,3)::"ENTER YOUR CHOICE"
    ACCEPT AT(23,21)VALIDATE(DIGIT)
    BEEP::CH=1 OR CH=2 OR CH=3 THEN 530
510 TO 760 ELSE CHA=30::::CHB=20::::GOTO 760
520 REM LASH CARD SCREEN
530 CALL CLEAR::::CALL SCREEN(12)::::FOR
    Z=1 TO 8::::CALL COLOR(Z,2,15)::::NEXT Z
    CALL HCHAR(1,1,104,768)
540 FOR Z=2 TO 5::::CALL HCHAR(Z,5,32,22
    ::::FOR Z=8 TO 16::::CALL HCHAR(Z,10,32,12)
    NEXT Z::::CALL HCHAR(6,6,96,22)::::CALL VCHAR(
    27,96,4)::::CALL HCHAR(17,11,96,12
    ))::::CALL VCHAR(9,22,96,8))
```

Continued

```
560 DISPLAY AT(3,4):"RIGHT:" :: DISPLAY AT(5,9):WR :: IF WR=0 THEN SCORE=SCORE-WRONG :: WRONG=0
570 DISPLAY AT(3,4):"WRONG:" :: DISPLAY AT(5,9):WRONG :: IF WRONG=0 THEN SCORE=SCORE-WRONG :: WRONG=0
580 RT=0 :: MRK=0 :: SC=0 :: ELSE SC=INT(RT/(RT+WR)*100+.5) :: SC=0 :: ELSE SC=INT(RT/(RT+WRONG)*100+.5)
590 DISPLAY AT(3,1):"SIZE(3):"SIZE(3) :: DISPLAY AT(5,1):"SIZE(4):"SIZE(4) :: DISPLAY AT(3,10):STR$(RT);"&" :: DISPLAY AT(5,10):STR$(SC)&"%"
600 A=INT(RND*11)+2 :: B=INT(RND*11)+2
610 A$=STR$(A):B$=STR$(B) :: C$=STR$(B)
620 REMALL HCHAR(12,32,5) :: CALL HCHAR(12,32,4)
630 CALL HCHAR(12,14,95,7) :: CALL HCHAR(12,32,4)
640 CALL SPRITE(#1,42,2,177,14) :: CALL SPRITE(#2,42,2,177,32)
650 REM CHECK ANSWER
660 IF RT=10 THEN FOR I=1 TO 10 :: CALL DELSPRITE(#I) :: NEXT I :: GOTO 700
670 REMEM=1
680 REM DISPLAY AT(24,7):"PRESS ANY KEY TO CONTINUE" :: CALL KEY(0,K,S) :: IF S=0 THEN 680
690 IF K=13 THEN 700 ELSE 680
700 RT=0 :: WR=0 :: FOR I=1 TO 10 :: CALL DELSPRITE(#I) :: NEXT I :: GOTO 100
710 REM DISPLAY AT(24,7):"PRESS ANY KEY TO CONTINUE" :: CALL KEY(0,K,S) :: IF S=0 THEN 710
720 REM DISPLAY AT(24,7):"PRESS ANY KEY TO CONTINUE" :: CALL KEY(0,K,S) :: IF S=0 THEN 720
730 REM DISPLAY AT(24,7):"PRESS ANY KEY TO CONTINUE" :: CALL KEY(0,K,S) :: IF S=0 THEN 730
740 CALL CLEAR :: GOTO 200
750 REM LONG DIVISION SCREEN
760 CALL CLEAR :: CALL SCREEN(2) :: FOR I=2 TO 23 :: CALL HCHAR(I,1,32,32) :: NEXT I :: RETURN
770 CALL HCHAR(11,2,96,28) :: CALL HCHAR(12,2,96,28) :: CALL HCHAR(13,2,96,28) :: RETURN
780 CALL DISPLAY AT(10,2):"2" :: CALL DISPLAY AT(11,1):"-" :: CALL DISPLAY AT(12,2):"0" :: RETURN
790 DISPLAY AT(1,7):"DIVISION TUTOR" :: RETURN
800 FOR Z=1 TO 7 :: CALL COLOR(Z,2,16) :: NEXT Z :: RETURN
810 CALL HCHAR(8,2,96,28) :: CALL HCHAR(9,2,96,28) :: CALL HCHAR(10,2,96,28) :: RETURN
820 CALL VCHAR(9,7,117,14) :: CALL HCHAR(23,8,119,18) :: RETURN
830 FOR Z=9 TO 22 :: CALL HCHAR(Z,8,58,2) :: NEXT Z :: RETURN
840 DISPLAY AT(2,9):"DIVIDE" :: DISPLAY AT(4,9):"MULTIPLY" :: DISPLAY AT(6,9):"SUBTRACT" :: RETURN
850 DISPLAY AT(5,9):"BRINGDOWN" :: DISPLAY AT(6,9):"REMAINDER" :: RETURN
860 REM SELECT(RND*CHAR(4)+2) :: B=INT(RND*CHAR(4)+2) :: IF A=B THEN 860 ELSE RETURN
870 A=INT(RND*CHAR(4)+2) :: B=INT(RND*CHAR(4)+2) :: IF A=B THEN 870 ELSE RETURN
880 CALL SPRITE(#1,42,2,14,95,LEN(B$)+2)
890 DISPLAY AT(13,12-LEN(A$)):A$&CHR$(4)&B$;
900 ROW=14 :: COL=13+LEN(B$)-LEN(STR$(I))
910 FOR PROB=1 TO LEN(B$)
920 CALL HCHAR(ARO,10,32) :: ARO=2 :: IF MRK=0 THEN PROB=COL-12
930 CALL HCHAR(ARO,10,129) :: IF MRK=0 THEN SD=INT(VAL(SEG$(B$,1,PROB))) ELSE SD=F
940 CALL SOUND(-1200,-7,0) :: GOSUB 1260
950 CALL CLEAR :: CALL KEY(0,K,S) :: IF S=0 THEN 950 ELSE RETURN
960 IF (MRK=0)*(D=0) THEN CALL CLEAR :: GOSUB 1460 :: GOTO 970
970 IF SD=D THEN 980 ELSE IF SD>D THEN 990 ELSE 1000
980 CALL HCHAR(ARO,10,32) :: ARO=ARO+1 :: DA$=STR$(D) :: IF (D=0)*(PROB=LEN(B$)) THEN ROW=ROW-1 :: GOTO 1160
990 CALL HCHAR(ARO,10,129) :: CT=0 :: GOTO 1160
1000 CALL HCHAR(ARO,10,129) :: CT=0 :: GOTO 1160
1010 CALL SOUND(200,-7,0) :: CALL HCHAR(ARO,10,32) :: ARO=ARO+1 :: DA$=STR$(D) :: CT=CT+1
1020 IF CT=3 THEN GOSUB 1320 :: ELSE IF CT=2 THEN GOSUB 1350 :: ELSE 1030
1030 CALL HCHAR(ROW+1,COL-LEN(DA$)+2,94,LEN(DA$)) :: CALL HCHAR(ARO,10,32)
1040 IF MRK=0 THEN ROW=ROW+2 :: IF PROB=1 THEN ELSE SD=VAL(SEG$(B$,1,PROB))
1050 CALL HCHAR(ARO,10,129,29) :: SP=LEN(STR$(D))+2 :: CALL SOUND(-LEN(STR$(D))*100,-7,0)
1060 CALL DELSPRITE(#1) :: CALL SOUND(-LEN(STR$(D))*100,-7,0) :: CALL DELSPRITE(#2) :: CALL SOUND(-LEN(STR$(D))*100,-7,0)
1070 IF SD<CT THEN GOSUB 1320 :: ELSE IF SD=CT THEN GOSUB 1350 :: ELSE 1080
1080 CALL HCHAR(9,9,58,15) :: F=S :: IF F<10 THEN S=S*10+BD
1090 CALL HCHAR(ARO,10,32) :: ARO=ARO+1 :: IF ARO=25 THEN ARO=6 :: CALL HCHAR(ARO,10,32)
1100 CALL HCHAR(ARO,10,129) :: CALL HCHAR(ARO,10,129)
1110 CALL VALIDATE(DIGIT)BEEP :: IF CT=2 THEN 1130 ELSE 1120
1120 CALL HCHAR(ROW,COL+2,130) :: CALL VALIDATE(DIGIT)BEEP
1130 CALL HCHAR(ROW-1,COL+4,131) :: CALL VALIDATE(DIGIT)BEEP
1140 IF F=S*10+BD THEN 1200 ELSE 1150
1150 ROB=ROB+1 :: IF ROB=LEN(B$)+1 THEN 1160 ELSE 1140
1160 REMAINDER
1170 DISPLAY AT(ROW,COL):"0" :: ROW=ROW-1 :: CT=0 :: GOTO 1120
1180 CALL HCHAR(ARO,10,32) :: ARO=6 :: CALL HCHAR(ARO,10,129) :: CALL HCHAR(ARO,10,129)
1190 IF CT=3 THEN CALL HCHAR(ROW-1,COL+4,131) :: CALL HCHAR(ROW,COL+4,131) :: GOTO 1200 ELSE CALL HCHAR(ROW,COL+4,131)
1200 CALL VALIDATE(DIGIT)BEEP :: IF CT=2 THEN 1210 ELSE 1220
1210 IF F=S*10+BD THEN 1220 ELSE GOSUB 1320 :: GOTO 1230
1220 CORRECT :: GOSUB 1470 :: GOSUB 1350 :: RESTORE 2000 :: FOR I=1 TO 5 :: READ S :: NEXT I :: CALL SOUND(S,S,5)
1230 DISPLAY AT(24,4):"PRESS ANY KEY TO CONTINUE" :: CALL KEY(0,K,S) :: IF S=0 THEN 1230
1240 IF K<>13 THEN 1240 ELSE 1250
1250 CALL HCHAR(Z,8,58,2) :: NEXT Z :: GOTO 870
1260 DISPLAY AT(9,9):SIZE(15) :: DISPLAY AT(9,9):SIZE(15) :: RETURN
1270 RETURN
1280 DISPLAY AT(9,9):SIZE(15) :: RETURN
1290 RETURN
1300 DISPLAY AT(9,9):SIZE(15) :: RETURN
1310 RETURN
1320 RT=RT+1 :: FOR I=1 TO 4 :: CALL SOUND(400,110,0) :: NEXT I :: RETURN
1330 CALL SOUND(400,220,0) :: DISPLAY AT(10,15):"WRONG" :: RETURN
1340 REMSWER IS "; :: CALL SOUND(-100,-7,0) :: NEXT I :: DISPLAY AT(12,1):"
1350 REMAINDER" :: RETURN
1360 DISPLAY AT(24,7):" THE ANSWER IS " :: CALL SOUND(-100,-7,0) :: NEXT I :: DISPLAY AT(12,1):"
1370 ON 42-INT(SD/A) GOTO 1380,1390,1400,1410,1420,1430,1440,1450
1380 D=INT(SD/A) :: DISPLAY AT(11,COL):SIZE(D) :: RETURN
1390 D=(SD-D*A)*10 :: DISPLAY AT(ROW,COL-LEN(DA$)):DA$ :: RETURN
1400 DISPLAY AT(ROW,COL-SP):SIZE(D) :: RETURN
1410 DISPLAY AT(11,COL):SIZE(D) :: RETURN
1420 IF D=0 THEN 1430 :: DISPLAY AT(10,COL+2):SIZE(D) :: RETURN
1430 DISPLAY AT(10,COL+2):SIZE(D) :: RETURN
1440 DISPLAY AT(13,3):"ERASE ALL BEEP" :: GOSUB 1460 :: RETURN
1450 DISPLAY AT(13,3):"ENTER ANSWER" :: GOSUB 1460 :: RETURN
1460 FOR Z=1 TO 8 :: CALL COLOR(Z,2,15) :: NEXT Z :: RETURN
1470 DATA 2,440,523,2,523,587,2,587,659,2,659,2,740,740,2,659
1480 DATA 587,2,659,2,740,2,740,2,659,659
1490 DATA 740,740
```


HCM

PEG JUMP


```
100 REM ************
110 REM * PEG JUMP *
120 REM ************
130 REM BY BOB STOFFERS
140 REM AND THE HCM STAFF
150 REM HOME COMPUTER MAGAZINE
160 REM VERSION 4.5.1
170 REM APPLE II FAMILY APPLESOFT
180 REM
190 HCOLOR= 3
200 TEXT : HOME
210 DIM PX(33),PY(33),PF(33),AS(62)
220 RESTORE
230 FOR I = 1 TO 33: READ PX(I),PY(I): NEXT
240 DATA 120,20,140,20,160,20,120,40,140,40,160,40,80,60,100,60,120,60,140,60,160,60,180,60,200,60,80,80,100,80,120,80,140,80,160,80,180,80,200,80
250 DATA 80,100,100,100,120,100,140,100,160,100,180,100,200,100,120,120,140,120,160,120,120,140,140,140,160,140
260 FOR I = 1 TO 62: READ AS(I): NEXT
270 DATA 15,17,28,16,21,23,7,21,17,15,24,22,26,24,21,23,8,22,29,17,33,25,22,24,31,33,18,30,4,16,33,25
280 DATA 6,18,24,26,13,11,26,12,18,6,27,13,16,18,3,11,1,3,18,6,13,11,10,12,3,11,12,10,5,17
290 FOR I = 1 TO 33:PF(I) = 1: NEXT :PF(17) = 0
300 FOR I = 0 TO 8
310 READ PLS$(I)
320 NEXT
330 DATA "JACKPOT!! YOU'RE THE GREATEST","SUPER!! BUT LAST PEG NOT IN CENTER","EXPERT!! ALMOST THE JACKPOT","PRO PLAYER STATUS","YOU'RE NO AMATEUR"
340 DATA "AVERAGE PLAYER LEVEL","NOT QUITE AVERAGE YET","GETTING CLOSE TO AVERAGE","JACKPOT!! IF I CAN DO IT SO CAN YOU"
350 POKE 33,40: HOME
360 VTAB 11: INVERSE : HTAB 15: PRINT ":       :": HTAB 15: PRINT " PEG JUMP ": HTAB 15: PRINT ":       :"
370 NORMAL
380 VTAB 20: HTAB 13: PRINT "PLEASE STAND BY"
390 GOSUB 1490
400 GOSUB 1180
410 GOSUB 1020
420 FOR PI = 230 TO 20 STEP - 10:LE = 10: GOSUB 1450: NEXT
430 REM ******* MAIN PROGRAM *******
440 PL = 0: FOR I = 1 TO 33: IF PF(I) THEN DRAW 11 AT PX(I),PY(I):PL = PL + 1
450 POKE 33,40: HOME
460 NEXT
470 VTAB 21: HTAB 23: PRINT "PEGS LEFT ";PL
480 HTAB 23: PRINT "PEGS REMOVED ";32 - PL
490 GOSUB 1510
500 POKE 33,20
510 INPUT "FROM ";FR$: IF LEN (FR$) > 2 OR FR$ = "" THEN 510
520 FOR Z = 1 TO LEN (FR$): IF ASC ( MID$ (FR$,Z,1)) < 48 OR ASC ( MID$ (FR$,Z,1)) > 57 THEN GOTO 510
530 NEXT
540 FR = VAL (FR$): IF FR = 88 THEN GOTO 940
550 IF FR = 99 THEN GOTO 220
560 INPUT "TO ";T$: IF LEN (T$) > 2 OR T$ = "" THEN 560
570 FOR Z = 1 TO LEN (T$): IF ASC ( MID$ (T$,Z,1)) < 48 OR ASC ( MID$ (T$,Z,1)) > 57 THEN GOTO 560
580 NEXT
590 T = VAL (T$): IF T < 1 OR FR < 1 OR T > 33 OR FR > 33 THEN GOTO 510
600 IF PF(FR) = 0 THEN GOTO 510
610 IF PF(T) = 1 THEN GOTO 510
620 GOSUB 630: GOTO 440
630 TX = PX(T):TY = PY(T):FX = PX(FR):FY = PY(FR)
640 IF TY < > FY THEN 740
650 IF TX + 40 < > FX AND FX + 40 < > TX THEN RETURN
660 IF TX + 40 = FX THEN XX = TX + 20:YY = TY
670 IF FX + 40 = TX THEN XX = FX + 20:YY = TY
680 FOR I = 1 TO 33: IF XX = PX(I) AND YY = PY(I) THEN 700
690 NEXT
700 PN = I: FOR I = 1 TO 1: NEXT
710 IF NOT PF(PN) THEN RETURN
720 PF(PN) = 0:PF(FR) = 0:PF(T) = 1
730 GOTO 830
740 IF TX < > FX THEN RETURN
750 IF TY + 40 < > FY AND FY + 40 < > TY THEN RETURN
760 IF TY + 40 = FY THEN XX = TX:YY = TY + 20
770 IF FY + 40 = TY THEN XX = TX:YY = FY + 20
780 FOR I = 1 TO 33: IF XX = PX(I) AND YY = PY(I) THEN 800
790 NEXT
800 PN = I: FOR I = 1 TO 1: NEXT
810 IF NOT PF(PN) THEN RETURN
820 PF(PN) = 0:PF(FR) = 0:PF(T) = 1
830 X1 = PX(FR):Y1 = PY(FR):X2 = PX(T):Y2 = PY(T)
840 XDRAW 11 AT X1,Y1
850 IF X1 = X2 THEN 890
860 FOR I = X1 TO X2 STEP ( SGN (X2 - X1)) * 3
870 XDRAW 11 AT I,Y1:PI = I + 30:LE = 6: GOSUB 1450: XDRAW 11 AT I,Y1: NEXT
880 GOTO 920
890 FOR I = Y1 TO Y2 STEP ( SGN (Y2 - Y1)) * 3
900 XDRAW 11 AT X1,I:PI = I + 40:LE = 6: GOSUB 1450: XDRAW 11 AT X1,I: NEXT
910 HCOLOR= 3
920 XDRAW 11 AT PX(T),PY(T)
930 FOR I = 1 TO 5: XDRAW 11 AT PX(PN),PY(PN):PI = I * 5 + 32:LE = 10: GOSUB 1450: NEXT : RETURN
940 REM ** AUTO SOLVE
950 FOR I = 1 TO 33:PF(I) = 1: NEXT :PF(17) = 0: GOSUB 1020
960 FOR I = 1 TO 33: IF PF(I) THEN DRAW 11 AT PX(I),PY(I)
970 NEXT
980 FOR AS = 1 TO 61 STEP 2
990 FR = AS(AS):T = AS(AS + 1): GOSUB 630
1000 NEXT
1010 POKE 33,40: HOME : VTAB 22: PRINT PLS$(8): FOR I = 1 TO 500: NEXT : GOTO 220
1020 HGR : FOR X = 70 TO 210 STEP 20
1030 HPLOT X,50 TO X,110: IF X > = 110 AND X < = 170 THEN HPLOT X,10 TO X,150
1040 NEXT
1050 FOR Y = 10 TO 150 STEP 20
1060 HPLOT 110,Y TO 170,Y: IF Y > = 50 AND Y < = 110 THEN HPLOT 70,Y TO 210,Y
1070 NEXT
1080 FOR I = 1 TO 33
1090 X = PX(I) - 8:Y = PY(I) - 4
1100 S1 = INT (I / 10)
1110 S2 = I - 10 * ( INT (I / 10))
1120 IF S2 = 0 THEN S2 = 10
1130 IF S1 THEN DRAW S1 AT X,Y: DRAW S2 : GOTO 1150
1140 DRAW S2 AT X,Y
1150 NEXT
1160 RETURN
1170 END
1180 FOR I = 24576 TO 24725
1190 READ A: POKE I,A: NEXT
1200 REM ** START OF TABLE
1210 DATA 12,0,32,0,40,0,53,0,65,0,75,0,86,0,96,0,108,0,119,0,129,0,140,0,159,0,161,0,5,0,5,0
1220 REM ** SHAPE #1 (1)
1230 DATA 41,229,36,103,149,138,09,0
1240 REM ** SHAPE #2 (2)
1250 DATA 45,45,193,219,12,5,193,28,191,78,137,10,0
1260 REM ** SHAPE #3 (3)
1270 DATA 45,12,220,45,193,28,63,150,74,73,0,0
1280 REM ** SHAPE #4 (4)
1290 DATA 9,36,36,86,31,39,77,146,9,0
1300 REM ** SHAPE #5 (5)
1310 DATA 45,5,193,28,63,36,45,181,82,1,0
1320 REM ** SHAPE #6 (6)
1330 DATA 41,12,28,247,36,12,109,146,10,0
1340 REM ** SHAPE #7 (7)
1350 DATA 33,5,193,5,193,197,63,183,82,73,1,0
1360 REM ** SHAPE #8 (8)
1370 DATA 41,12,28,247,4,193,12,173,142,10,0
1380 REM ** SHAPE #9 (9)
1390 DATA 41,12,36,28,191,14,173,74,0
1400 REM ** SHAPE #10 (0)
1410 DATA 00,41,5,193,36,28,191,54,78,73,00,00
1420 REM ** SHAPE #11 (PEG)
1430 DATA 60,54,53,55,53,79,36,36,36,0
1440 POKE 232,0: POKE 233,96: SCALE= 1: ROT= 0: RETURN
1450 POKE 0,255 - PI
1460 POKE 1,LE
1470 CALL 771
1480 RETURN
```

Continued

PEG JUMP *Continued*

APPLE // Family

```
1490 POKE 771,173: POKE 772,48: POKE 773
     ,192: POKE 774,136: POKE 775,208: P
     OKE 776,4: POKE 777,198: POKE 778,1
     : POKE 779,240: POKE 781,202: POKE 782,
1500 POKE 780,8: POKE 783,246: POKE 784,166: POKE 785,0: POKE 786,76: POKE 787,3:
     POKE 768,3: POKE 789,96: RETURN
1510 IF PL > 7 THEN RETURN
1520 POKE 33,20
1530 IF PL > 1 THEN 1560
1540 IF PF ( 17) POKE 33,40: HOME :
     VTAB 23: PRINT PLS$(0): GOTO 1570
1550 PRINT PLS$(1): GOTO 1570
1560 HOME : VTAB 23: PRINT PLS$(PL): RET
     URN
1570 FOR PI = 200 TO 40 STEP - 3:LE = 4
     : GOSUB 1450: NEXT : GOTO 220
```

HCM

PEG JUMP

COMMODORE 64

```
100 REM ********** 
110 REM * PEG JUMP *
120 REM **********
130 REM BY BOB STOFFERS
140 REM AND THE HCM STAFF
150 REM HOME COMPUTER MAGAZINE
160 REM VERSION 4.5.1
170 REM C-64 BASIC
180 REM
190 PRINT" SHIFT CLR"
200 POKE 53280,0:POKE 53281,0:POKE 657,
    128
210 REM DISPLAY THE TITLE SCREEN
220 PRINT:PRINT" CTRL BLU"
230 PRINT" CTRL RVSON SHIFT £
    £ SHIFT £
240 PRINT" CTRL RVSON CTRL RVSOF
    F CTRL RVSON CTRL RVSOF
    F CTRL RVSON CTRL RVSOF
    3CMDR IMDR
250 PRINT" CTRL RVSON CTRL RV
    OFF SHIFT £ CTRL RVSON CTRL RVS
    TRL RVSOFF CTRL RVSON CTRL RVS
    OFF CMDR K"
260 PRINT" CTRL RVSON CTRL RVSOF
    F CMDR K CTRL RVSON CTRL RVSOF
    RL RVSOF CTRL RVSON CTRL RVSO
    FF CMDR K CMDR LCTRL RVSON
270 PRINT" CTRL RVSON CTRL RVSOF
    F CMDR K CTRL RVSON CTRL RVSOF
    CTRL RVSOF 3CMDR ICTRL RVSOF
    L RVSOFF SHIFT £ CTRL RVSOF
280 PRINT" CTRL RVSON CTRL RVSOF
    F CMDR K CTRL RVSON CTRL RVSOF
    L RVSOFF SHIFT £ CTRL RVSON
290 PRINT:PRINT:PRINT" CTRL GRN"
300 PRINT" CMDR LRVSOF
    N CTRL RVSOF
    T £ CMDR LCTRL RVSOF
    ON CMDR K CTRL RVSOF
    L "RVSOF
310 PRINT" CMDR LCTRL RVSOF
    N CTRL RVSOF
    R LCTRL RVSOF
    K CTRL RVSOF
    CTRL RVSOF
    VSOFF CMDR K
320 PRINT" CMDR LCTRL RVSOF
    N CTRL RVSOF
    R LCTRL RVSOF
    K CTRL RVSOF
    CTRL RVSOF
    L RVSOFF SHIFT £
330 PRINT" CTRL RVSON SHIFT £
    CMDR LCTRL RVSOF
    RVSON CTRL RVSOF
    CMDR LCTRL RVSOF
    CMDR KCTRL RVSOF
    CTRL RVSOF
    RVSON CTRL RVSOF
    CMDR LCTRL RVSOF
    CMDR KCTRL RVSOF
    CTRL RVSOF
    RVSON CTRL RVSOF
340 PRINT" CTRL RVSON CTRL R
    VSOFF CTRL RVSON CTRL R
    VSOFF CTRL RVSON CTRL RVSOF
    CTRL RVSON CTRL RVSOF
    RVSON CTRL RVSOF
350 PRINT" CTRL RVSON SHIFT £
    CTRL RVSOF SHIFT £ CTRL RVSON
    CTRL RVSOF SHIFT £ CTRL RVSOF
    SHIFT £ CTRL RVSOF
360 PRINT:PRINT:PRINTTAB(5)" CTRL CYN*
    -*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-
370 GOSUB 1480
380 REM STORE SCREEN MESSAGES IN THE PL
    RS$ ARRAY BY -*-*-*-*-*-*-*-*-*-*-
390 FOR I=0 TO 8:FOR J=1 TO 4
400 READ PLS$(I,J)
410 NEXT:NEXT:INITIALIZE GAME VARIABLES
420 REM
430 DIM RC(33,2)
440 FOR I=1 TO 33:READ RC(I,1),RC(I,2):
    NEXT
450 D1=PEEK(65):D2=PEEK(66)
460 REM DISPLAY GAME SCREEN
470 PL=32
480 PRINT" SHIFT CLR":PRINT" CTRL WHT
    PEGS":TAB(32)"PEGS"
490 PRINT"  LEFT"TAB(13)" CTRL BLU CTRL
    RVSON CTRL WHT REMOVED"
500 PRINT PL" CTRL BLU CTRL R
    VSON SHIFT 1 SHIFT 2 SH
    IFT"3" CTRL WHT CTRL RVSOFF
    0
510 PRINTTAB(13)" CTRL BLU CTRL RVSON
520 PRINTTAB(13)" CTRL RVSON
530 PRINTTAB(13)" CTRL RVSON SHIFT
    4 SHIFT 5 SHIFT 6 "
540 PRINTTAB(13)" CTRL RVSON
550 PRINT" CTRL RVSON "
560 PRINT" CTRL RVSON SHIFT
    8 SHIFT 9 SHIFT
    10 SHIFT 11 SHIFT 12 SHIF
    T 13"
570 PRINT" CTRL RVSON "
580 PRINT" CTRL RVSON "
590 PRINT" CTRL RVSON SHIFT 14
    15 SHIFT 16 SHIFT Q
    17 SHIFT 18 SHIFT 19 SHIF
    T 20"
600 PRINT" CTRL RVSON "
610 PRINT" CTRL RVSON "
620 PRINT" CTRL RVSON SHIFT 21
    22 SHIFT 23 SHIFT
    24 SHIFT 25 SHIFT 26 SHIF
    T 27"
630 PRINT" CTRL RVSON "
640 PRINTTAB(13)" CTRL RVSON
650 PRINTTAB(13)" CTRL RVSON SHIFT
    28 SHIFT 29 SHIFT 30 "
660 PRINTTAB(13)" CTRL RVSON
670 PRINTTAB(13)" CTRL RVSON
680 PRINTTAB(13)" CTRL RVSON SHIFT
    31 SHIFT 32 SHIFT 33 "
690 PRINTTAB(13)" CTRL RVSON
700 PRINT;TAB(53)" CTRL WHT FROM:  TO:
710 FOR I=1TO33:IF I=17 THEN730
720 POKE (RC(I,1)*40+RC(I,2)+55296),3
730 NEXT
740 REM GET INPUT FOR MOVE AND CHECK FO
    R A LEGAL MOVE
750 IF F88=1 AND PL=1 THEN1470
760 IF F88=1 THEN1450
770 IF PLX=24 THEN1100
780 T=0:PLX=24:Y=18:GOSUB1710:F=K
790 IF F=99 THEN460
800 IF F=88 THEN F88=1:POKE 65,D1:POKE
    66,D2:GOTO460
810 IF F>33 OR F<1 THEN1280
820 X=24:Y=25:GOSUB1740:T=K
830 IF MSG=1 THEN GOSUB1390
840 IF T>33 OR T<1 THEN1280
850 IF RC(F,1)=RC(T,1) THEN MD=5:DF=RC(
    F,2)-RC(T,2):GOTO880
860 IF RC(F,2)=RC(T,2) THEN MD=3:DF=RC(
    F,1)-RC(T,1):GOTO880
870 GOTO1280
880 IF ABS(DF)<>(MD*2) THEN1280
890 DF=DF/2:IF MD=3 THEN DF=DF*40
900 IF PEEK(RC(F,1)*40+RC(F,2)+1024)<>2
    PEEK(RC(F,1)*40+RC(F,2)-DF+1024)
910 IF 2PEEK(RC(T,1)*40+RC(T,2)+1024)<>2
    THEN1280
920 IF PEEK(RC(T,1)*40+RC(T,2)+1024)<>2
    THEN1280
930 REM MOVE PEG FROM START HOLE TO END
    HOLE AND REMOVE PEG FROM MIDDLE
940 POKE L1+24,15:POKE L1+5,68:POKE L1+
    6,68:RC(F,1):Y=RC(F,2):GOSUB1820:PRINT
950 X=RC(F,1):RC(T,1):RC(F,2):GOSUB1820
    :PRINT" CTRL BLU CTRL RVSON
960 PL=PL-1:SP=40:IF MD=5 THEN SP=1
970 IF DF>0 THEN SP=SP*-1
980 FOR I=RC(F,1)*40+RC(F,2)+SP+1024 TO
    RC(T,1)*40+RC(T,2)-SP+1024 STEP SP
990 POKE L1+1,I*10AND255:POKE L1,I*10AN
    D255:POKE D2,55
1000 J=PEEK(I):IF J=215 THEN J=209
```

Continued

PROGRAM LISTING

PEG JUMP *Continued*

COMMODORE 64

```
1010 POKE (I),215:POKE (I+L1),6
1020 POKE L1+4,17
1030 FOR M=1 TO 125:NEXT
1040 POKE L1+4,16
1050 POKE (I),J:POKE (I+54272),6
1060 NEXT:FOR I=L1 TO L1+24:POKEI,0:NEXT
1070 X=RC(T,1):Y=RC(T,2):GOSUB1820:PRINT"[CTRL CYN][CTRL RVSON][SHIFT W][CTRL WHT]"
1080 X=3:Y=0:GOSUB1820:PRINTPL"[SHIFT CRSR LEFT]"TAB(31)32-PL"[SHIFT CRSR LEFT]"
1090 GOTO740
1100 REM DISPLAY MESSAGES WHEN PLAYER GETS CLOSE TO END OF GAME
1110 PLS=PL
1120 IF PLS=1 AND PEEK(1523)=215 THEN PLS=0
1130 POKE L1+1,34:POKE L1,75:POKE L1+6,130:POKE L1+5,68:POKE L1+24,15
1140 POKE L1+4,17:FOR I=1 TO 100:NEXT:POKE L1+4,16
1150 FOR I=1 TO 25:NEXT
1160 POKE L1+1,68:POKE L1,149:POKE L1+4,17:FOR I=1 TO 200:NEXT:POKE L1+4,16
1170 FOR I=L1 TO L1+24:POKE I,0:NEXT
1180 X=18:Y=0:GOSUB1820
1190 FOR I=1 TO 4
1200 PRINTTAB(29)PLS$(PLS,I);
1210 FOR J=1 TO 10-LEN(PLS$(PLS,I)):
1220 PRINT" ";
1230 NEXT:PRINT:NEXT
1240 IF PLS=0 THEN GOSUB1480
1250 IF (F88<>1 AND PL<>1) OR (PEEK(1523)<>215) THEN780
1260 F88=0:GOSUB1480
1270 GOTO740
1280 REM DISPLAY ILLEGAL MOVE MESSAGE
1290 GOSUB1390
1300 FOR I=1 TO 15:POKE L1+24,15
1310 FOR J=1 TO 3:NEXT:POKE L1+24,0
1320 FOR M=1 TO 3:NEXT:NEXT
1330 MSG=1:X=18:Y=1:GOSUB1820
1340 PRINT"FROM:"F:X=20:GOSUB1820
1350 IF T=0 THEN1370
1360 PRINT"TO:"T:X=22:GOSUB1820
1370 PRINT"ILLEGAL":PRINT" MOVE"
1380 GOTO740
1390 REM CLEAR MESSAGE AREAS OF THE SCREEN
1400 MSG=0:X=18:Y=MSG:GOSUB1820
1410 FOR I=1 TO 6:
1420 PRINT"           "TAB(29)"
      "
1430 NEXT
1440 PRINT"           "TAB(29)"
      ";:RETURN
1450 READ F,T:X=24:Y=17:GOSUB1820:PRINT"[CTRL WHT]"F"[SHIFT CRSR LEFT]  ";:GOSUB1820:PRINT":";
1460 Y=24:GOSUB1820:PRINTT"[SHIFT CRSR LEFT]";:GOSUB1820:PRINT":";:GOTO850
1470 PLS=8:GOTO1130
1480 REM PLAY THE THEME MUSIC
1490 RESTORE
1500 L1=54272:L2=54279:L3=54286
1510 H1=L1+1:H2=L2+1:H3=L3+1
1520 V1=L1+4:V2=L2+4:V3=L3+4
1530 POKE 54296,15
1540 POKE V1+1,9:POKE V2+2,0
1550 POKE V2+1,36:POKE V2+2,36
1560 POKE V3+1,18:POKE V3+2,170
1570 T=TI
1580 POKE V1,16:POKE V2,32:POKE V3,16
1590 READ S:IF S=0 GOTO1680
1600 READ X1,Y1,X2,Y2,X3,Y3
1610 POKE 53280,X1
1620 IF X1 THEN POKE H1,X1:POKE L1,Y1:POKE V1,17
1630 IF X2 THEN POKE H2,X2:POKE L2,Y2:POKE V2,33
1640 IF X3 THEN POKE H3,X3:POKE L3,Y3:POKE V3,17
1650 T=T+S
1660 IF T>TI GOTO1660
1670 GOTO1580
1680 FORJ=L1 TO 54296:POKE J,0:NEXT J
1690 POKE 53280,0
1700 RETURN
1710 REM INPUT ROUTINE
1720 POKE 781,24:POKE 782,13:POKE783,0:SYS 65520
1730 PRINT"[CTRL WHT]FROM:      TO:     ";
1740 GOSUB1820:POKE 198,0:C=0:K=C
1750 PRINT"[CMDR @][SHIFT CRSR LEFT]";
1760 GET K$:IF K$="" THEN1760
1770 IF C AND ASC(K$)=13 THEN POKEL1+24,15:PRINT" ";:K=K/10:POKE L1+24,0:RETURN
1780 IF ASC(K$)<48 OR ASC(K$)>57 THEN1760
1790 POKE L1+24,15:C=C+1:PRINTK$;:POKE L1+24,0
1800 IF C=1 THEN K=VAL(K$)*10:GOTO1750
1810 K=K+VAL(K$):RETURN
1820 REM ROUTINE TO PLACE THE CURSOR ANYWHERE ON THE SCREEN
1830 POKE 781,X:POKE 782,Y:POKE 783,0:SYS 65520:RETURN
1840 REM THEME MUSIC DATA
1850 DATA 20,0,0,0,0,0,0
1860 DATA 20,0,0,21,154,4,73
1870 DATA 20,0,0,25,177,6,108
1880 DATA 20,34,75,21,154,8,147
1890 DATA 10,43,100,0,0,0,0
1900 DATA 10,38,126,25,177,0,0
1910 DATA 20,0,0,19,63,9,159
1920 DATA 20,0,0,25,177,0,0
1930 DATA 20,51,97,22,227,6,108
1940 DATA 10,43,52,0,0,0,0
1950 DATA 10,51,97,25,177,7,53
1960 DATA 20,0,0,17,37,8,23
1970 DATA 20,0,0,25,177,0,0
1980 DATA 20,68,149,21,154,8,147
1990 DATA 10,51,97,0,0,0,0
2000 DATA 10,68,149,25,177,0,0
2010 DATA 40,0,0,17,37,6,108
2020 DATA 40,86,105,21,154,8,147
2030 DATA 0
2040 REM MESSAGE DATA
2050 DATA JACKPOT!!,YOU'RE,THE,GREATEST,SUPER!!,BUT LAST,PEG NOT,IN CENTER
2060 DATA EXPERT!!,ALMOST,THE,JACKPOT,PRO,PLAYER,STATUS," ",YOU'RE,NO,AMATEUR
2070 DATA " ",AVERAGE,PLAYER,LEVEL," ",NOT,QUITE,AVERAGE,YET
2080 DATA GETTING,CLOSE,TO,AVERAGE,JACKPOT!!
2090 DATA IF I CAN,DO IT SO,CAN YOU!!
2100 REM DATA FOR PEG LOCATIONS
2110 DATA 3,14,3,19,3,24,6,14,6,19,6,24
2120 DATA 9,4,9,9,9,14,9,19,9,24,9,29,9,34
2130 DATA 12,4,12,9,12,14,12,19,12,24,12,29,12,34
2140 DATA 15,4,15,9,15,14,15,19,15,24,15,29,15,34
2150 DATA 18,14,18,19,18,24,21,14,21,19,21,24
2160 REM DATA FOR AUTO SOLVE MODE
2170 DATA 15,17,28,16,21,23,7,21,17,15,24,22,26,24,21,23,8,22,29,17,33,25
2180 DATA 22,24,31,33,18,30,4,16,33,25,6,18,24,26,13,11,26,12,18,6,27,13
2190 DATA 16,18,3,11,1,3,18,6,13,11,10,12,3,11,12,10,5,17
```

HCM

PEG JUMP

IBM PC & IBM PCjr

```
100 '  ***********
110 '  * PEG JUMP *
120 '  ***********
130 '  BY BOB STOFFERS
140 '  HOME COMPUTER MAGAZINE
150 '  VERSION 4.5.1
160 '  IBM PCjr CARTRIDGE BASIC or
170 '  IBM PC BASICA WITH
180 '  COLOR/GRAPHICS MONITOR ADAPTER
190 '  AND COLOR MONITOR
200 '
210 '  TITLE SCREEN
220 KEY OFF:SCREEN 0:WIDTH 40:COLOR 11,1,9:CLS:LOCATE 11,12:PRINT "P E G  J U M P":LOCATE 22,7:PRINT "PRESS [ENTER] TO CONTINUE":PLAY "MF"
230 '  INITIALIZE THE PROGRAM
240 A$=INKEY$:IF A$<>CHR$(13) THEN 240 ELSE CLS:SCREEN 1:COLOR 1,1:DEFINT A-Z:DIM P(17),H(7),PP(33,2),PLS$(8,4),CRS(3)
250 DEF FNMIN(V1,V2)=((V1>V2)*V2*(-1))+((V2>V1)*V1*(-1))
260 CRS(0)=223:CRS(1)=221:CRS(2)=220:CRS(3)=222
270 DRAW "BM10,20C2BD4F5ERFRULU2EH5G4BR4P2,2BM30,20C3BLBU2R2FD2GL2HU2ERBD3P3,3":GET(10,20)-(19,29),P:GET(28,18)-(32,22),H:GOSUB 290:GOTO 380
280 '  SUBROUTINE TO DRAW BOARD
290 CLS:DRAW "BM160,0C2R48D48R64D80L64D48L96U48L64U80R64U48R48BD5P3,2BU5C2BR48F16D32BH4P3,2C2BF4BR48F16D80H16BR4P3,2C2BL4BF16L64H16BR16BD8P3,2C2BD8D48H16BR8P3,2C2BR8BD16L96H16BR16BD8P3,2C2BL16BU40L48H16BR8BD4P3,2"
300 FOR Z=128 TO 192 STEP 32:LINE (Z,16)-(Z,160),2:NEXT:FOR Z=64 TO 112 STEP 24:LINE (64,Z)-(256,Z),2:NEXT:FOR Z=16 TO 160 STEP 24:LINE (128,Z)-(192,Z),2:NEXT:FOR Z=64 TO 256 STEP 32:LINE (Z,64)-(Z,112),2:NEXT
310 RESTORE 780:FOR Z=1 TO 2:FOR ZZ=1 TO 33:READ PP(ZZ,Z):NEXT:NEXT
320 FOR Z=0 TO 9:READ N$(Z):NEXT
330 FOR Z=0 TO 8:FOR ZZ=1 TO 4:READ PLS$(Z,ZZ):NEXT:NEXT
340 FOR Z=1 TO 33:PUT(PP(Z,1)-2,PP(Z,2)-2),H,PRESET:GOSUB 550:IF Z<>17 THEN PUT(PP(Z,1)-11,PP(Z,2)-11),P
```

Continued

PROGRAM LISTING

PEG JUMP *Continued*

IBM PC & IBM PCjr

```
350 NEXT:LOCATE 1,2:PRINT "PEGS":PRINT "  LEFT":PRINT "    32":LOCATE 1,34:PRINT "PEGS":LOCATE 2,32:PRINT "REMOVED":LOCATE 3,36:PRINT "0":LOCATE 20,1:PRINT "FROM:":PRINT:PRINT "   TO:":PL=32:PR=0
360 RETURN
370 ' PLAY THE GAME
380 IF F88=1 THEN GOSUB 680:GOTO 400 ELSE YC=20:GOSUB 570:F=VAL(A$):IF F=88 THEN GOSUB 650:GOTO 380 ELSE IF F=99 THEN GOSUB 290:GOTO 380 ELSE IF F<1 OR F>33 THEN GOSUB 700:GOTO 380 ELSE IF POINT(PP(F,1)-5,PP(F,2)-5)=3 THEN GOSUB 710:GOTO 380
390 YC=22:GOSUB 570:T=VAL(A$):IF T<1 OR T>33 THEN GOSUB 700:GOTO 380 ELSE IF POINT(PP(T,1)-5,PP(T,2)-5)=1 THEN GOSUB 720:GOTO 380
400 IF PP(F,1)<>PP(T,1) OR ABS(PP(F,2)-PP(T,2))<>48 THEN 420
410 MR=PP(F,1)-5:MC=FNMIN(PP(F,2),PP(T,2))+19:IF POINT(MR,MC)<>1 THEN GOSUB 740:GOTO 380 ELSE GOSUB 500:GOTO 440
420 IF PP(F,2)<>PP(T,2) OR ABS(PP(F,1)-PP(T,1))<>64 THEN GOSUB 730:GOTO 380
430 MC=PP(F,2)-5:MR=FNMIN(PP(F,1),PP(T,1))+27:IF POINT(MR,MC)<>1 THEN GOSUB 740:GOTO 380 ELSE GOSUB 460
440 GOSUB 750:GOSUB 760:GOTO 380
450 ' VERTICAL PEG JUMP
460 PUT(PP(F,1)-11,PP(F,2)-11),P
470 FOR Z=PP(F,1)-11 TO PP(T,1)-11 STEP SGN(PP(T,1)-PP(F,1))*2:PUT(Z,PP(F,2)-11),P:SOUND Z*5,1:PUT(Z,PP(F,2)-11),P:NEXT
480 PUT(PP(T,1)-11,PP(T,2)-11),P:PL=PL-1:PR=PR+1:PUT(PP(F,1)-11+(32*SGN(PP(T,1)-PP(F,1))),PP(F,2)-11),P:SOUND 110,3:GOTO 530
490 ' HORIZONTAL PEG JUMP
500 PUT(PP(F,1)-11,PP(F,2)-11),P
510 FOR Z=PP(F,2)-11 TO PP(T,2)-11 STEP SGN(PP(T,2)-PP(F,2))*2:PUT(PP(F,1)-11,Z),P:SOUND Z*5+40,1:PUT(PP(F,1)-11,Z),P:NEXT
520 PUT(PP(T,1)-11,PP(T,2)-11),P:PL=PL-1:PR=PR+1:PUT(PP(F,1)-11,PP(F,2)-11+(24*SGN(PP(T,2)-PP(F,2)))),P:SOUND 110,3
530 LOCATE 3,2:PRINT "   ":LOCATE 3,2:PRINT PL:LOCATE 3,36:PRINT "   ":LOCATE 3,36:PRINT PR:IF PL<8 THEN GOSUB 610:RETURN ELSE RETURN
540 ' SUBROUTINE TO DRAW SMALL NUMBERS
550 FOR ZZ=1 TO LEN(STR$(Z))-1:C=VAL(MID$(STR$(Z),ZZ+1,1)):PSET(PP(Z,1)+ZZ*5,PP(Z,2)+2):DRAW "C0;XN$(C);":NEXT:RETURN
560 ' KEY INPUT ROUTINE
570 DEF SEG=0:POKE 1050,PEEK(1052):A$="":FOR Z=0 TO 1
580 LOCATE YC,6+Z:CR=ABS((CR+1)*(CR<3)):PRINT CHR$(CRS(CR));CHR$(29);:K$=INKEY$:IF K$="" THEN 580 ELSE IF A$<>"" AND K$=CHR$(13) THEN SOUND 660,1:PRINT " ":RETURN ELSE IF K$<"0" OR K$>"9" THEN SOUND 110,3:GOTO 580 ELSE PRINT K$;:A$=A$+K$:SOUND 660,1
590 NEXT:RETURN
600 ' DISPLAY GETTING CLOSE MESSAGES
610 PLS=PL:IF PL=1 THEN CP=POINT(PP(17,1)-5,PP(17,2)-5):IF CP=1 THEN PLS=0:PLAY "L32O2CDEFGABO3CDEFGABO4CDEFGABO2L8DDDCCCEEE":IF F88=1 THEN PLS=8
620 FOR Z=1 TO 4:LOCATE Z+19,30:PRINT PLS$(PLS,Z):NEXT:PLAY "L16T200O2CDEDEFEFGFGAGABP1":FOR Z=20 TO 23:LOCATE Z,29:PRINT "          ";:NEXT:IF PL>1 THEN RETURN ELSE LOCATE 20,1:PRINT "PRESS 99 ":PRINT:PRINT "TO RESTART"
630 F88=0:FOR TD=1 TO 6000:NEXT:GOSUB 750:RETURN
640 ' SET UP FOR AUTO SOLVE MODE
650 GOSUB 290:RESTORE 850:F88=1:RETURN
660 ' SUBROUTINE TO READ AUTO SOLVE
670 ' MODE & DISPLAY MOVE
680 READ F,T:LOCATE 20,6:PRINT F:LOCATE 22,6:PRINT T:RETURN
690 ' ERROR MESSAGES
700 SOUND 110,5:GOSUB 750:LOCATE 20,1:PRINT "BAD VALUE":PRINT "PEGS ARE":PRINT "FROM 1 TO 33":PRINT "OR 88, OR 99":FOR TD=1 TO 3000:NEXT:GOSUB 750:GOSUB 760:RETURN
710 SOUND 110,5:GOSUB 750:LOCATE 20,1:PRINT "THERE IS":PRINT "NO PEG IN":PRINT "THAT HOLE":FOR TD=1 TO 3000:NEXT:GOSUB 750:GOSUB 760:RETURN
720 SOUND 110,5:GOSUB 750:LOCATE 20,1:PRINT "THERE IS":PRINT "ALREADY A":PRINT "PEG IN THE":PRINT "'TO' HOLE":FOR TD=1 TO 3000:NEXT:GOSUB 750:GOSUB 760:RETURN
730 SOUND 110,5:GOSUB 750:LOCATE 20,1:PRINT "ILLEGAL MOVE":PRINT "YOU MUST":PRINT "JUMP 1 PEG":PRINT "NOT DIAGONAL":FOR TD=1 TO 3000:NEXT:GOSUB 750:GOSUB 760:RETURN
740 SOUND 110,5:GOSUB 750:LOCATE 20,1:PRINT "BAD JUMP":PRINT "YOU MUST":PRINT "JUMP OVER":PRINT "A PEG":FOR TD=1 TO 3000:NEXT:GOSUB 750:GOSUB 760:RETURN
750 FOR ER=20 TO 23:LOCATE ER,1:PRINT "                    ":NEXT:LOCATE 20,1:RETURN
760 LOCATE 20,1:PRINT "FROM:":PRINT:PRINT "   TO:":RETURN
770 ' SCREEN POSITION FOR EACH HOLE
780 DATA 128,160,192,128,160,192,64,96,128,160,192,224,256,64,96,128,160,192,224,256,64,96,128,160,192,224,256,128,160,192,128,160,192
790 DATA 16,16,16,40,40,40,64,64,64,64,64,64,64,88,88,88,88,88,88,88,112,112,112,112,112,112,112,136,136,136,160,160,160
800 ' SMALL NUMBERS FOR THE BOARD
810 DATA "BRFD2GHU2E","BRGRD3NLR","BDEFDG2R2","RFGNLFGL","D2R2NU2D2","NR2DFRD2L2","BRGD2FEHL","R2D4","BRFGFGHEHE","R2D2ND2L2U2"
820 ' GAME STATUS MESSAGES
830 DATA JACKPOT!!,YOU'RE,THE,GREATEST,SUPER!!,BUT LAST,PEG NOT,IN CENTER,EXPERT!!,ALMOST,THE,JACKPOT,PRO,PLAYER,STATUS," ",YOU'RE,NO,AMATEUR," ",AVERAGE,PLAYER,LEVEL," ",NOT,QUITE,AVERAGE,AVERAGE,YET,GETTING,CLOSE,TO,AVERAGE,JACKPOT!!,IF I CAN,DO IT SO,CAN YOU!
840 ' AUTO SOLVE DATA
850 DATA 15,17,28,16,21,23,7,21,17,15,24,22,26,24,21,23,8,22,29,17,33,25,22,24,31,33,18,30,4,16,33,25,6,18,24,26,13,11,26,12,18,6,27,13,16,18,3,11,1,3,18,6,13,11,10,12,3,11,12,10,5,17
```

HCM

PROGRAM LISTING

PEG JUMP

TI-99/4A

```
100 REM ************
110 REM * PEG JUMP *
120 REM ************
130 REM BY BOB STOFFERS
140 REM AND THE HCM STAFF
150 REM HOME COMPUTER MAGAZINE
160 REM VERSION 4.5.1
170 REM TI BASIC
180 REM
190 CALL CLEAR
200 DEF MN=((V1>V2)*V2*(-1))+((V1<V2)*V1*(-1))
210 DEF MX=((V1<V2)*V2*(-1))+((V1>V2)*V1*(-1))
220 DIM RC(33,2),PLS$(8,4)
230 FOR I=2 TO 8
240 READ A,B
250 CALL COLOR(I,A,B)
260 NEXT I
270 DATA 7,7,14,14,5,5,8,1,8,1,8,1,8,1
280 CALL SCREEN(1)
290 FOR I=0 TO 8
300 FOR J=1 TO 4
310 READ PLS$(I,J)
320 NEXT J
330 NEXT I
340 DATA JACKPOT!!,YOU'RE,THE,GREATEST,SUPER!!,BUT LAST,PEG NOT,IN CENTER,EXPERT!!,ALMOST,THE,JACKPOT
350 DATA PRO,PLAYER,STATUS," ",YOU'RE,NO,AMATEUR," ",AVERAGE,PLAYER,LEVEL," ",NOT,QUITE,AVERAGE,YET
360 DATA GETTING,CLOSE,TO,AVERAGE,JACKPOT!!,IF I CAN,DO IT SO,CAN YOU!!
370 CALL HCHAR(1,1,40,256)
380 CALL HCHAR(9,1,48,256)
390 CALL HCHAR(17,1,56,256)
400 A$="P E G   J U M P"
410 X=12
420 Y=10
430 GOSUB 2510
440 GOSUB 2410
450 CALL CLEAR
460 A$="STAND BY PLEASE"
470 X=12
480 Y=7
490 GOSUB 2510
500 RESTORE 2690
510 FOR I=1 TO 33
520 READ RC(I,1),RC(I,2)
530 NEXT I
540 FOR I=90 TO 132
```

Continued

PEG JUMP *Continued*


```
550 READ A$
560 CALL HCHAR(I,A$)
570 NEXT I
580 CALL HCHAR(62,"0101010101010101FF")
590 CALL HCHAR(63,"80808080808080FF")
600 CALL HCHAR(136,"1C3E7F7F3E1C1C1C")
610 CALL COLOR(I,16,1)
620 NEXT I
630 CALL COLOR(I,2,16)
640 CALL COLOR(I,7,16)
650 CALL CLEAR
660 CALL SCREEN(5)
670 RANDOMIZE
680 DIM A(5),B(5),C(5),D(5)
690 FOR I=1 TO 5
700 READ A(I),B(I),C(I),D(I)
710 NEXT I
720 DATA 7,6,0,8,5,6,0,8,4,6,0,8,3,6,0,8,2,6,0,8
730 DATA 7,5,0,8,5,5,0,8,4,5,0,8,3,5,0,8,2,5,0,8
740 DATA 7,4,0,8,5,4,0,8,4,4,0,8,3,4,0,8,2,4,0,8
750 DATA 7,3,0,8,5,3,0,8,4,3,0,8,3,3,0,8,2,3,0,8
760 DATA 7,2,0,8,5,2,0,8,4,2,0,8,3,2,0,8,2,2,0,8
770 CALL CLEAR
780 CALL HCHAR(6,8,32,24)
790 CALL HCHAR(7,8,32,24)
800 CALL HCHAR(8,8,32,24)
810 CALL HCHAR(9,8,32,24)
820 CALL HCHAR(10,8,32,24)
830 CALL HCHAR(11,8,32,24)
840 CALL HCHAR(12,8,32,24)
850 CALL HCHAR(13,8,32,24)
860 CALL HCHAR(14,8,32,24)
870 CALL HCHAR(15,8,32,24)
880 CALL HCHAR(16,8,32,24)
890 CALL HCHAR(17,8,32,24)
900 CALL HCHAR(18,8,32,24)
910 CALL HCHAR(19,8,32,24)
920 CALL HCHAR(20,8,32,24)
930 CALL HCHAR(21,8,32,24)
940 CALL HCHAR(22,8,32,24)
950 CALL HCHAR(23,8,32,24)
960 CALL HCHAR(R,C(I,1),-1,RC(I,2),-1,I+99)
970 NEXT I
980 A$="FROM:  TO:"
990 X=24
1000 Y=9
1010 GOSUB 1280
1020 RESTORE 2850
1030 FOR I=0 TO 4
1040 READ X,Y
1050 GOSUB 1280
1060 NEXT I
1070 L=0
1080 FOR I=1 TO 33
1090 GOSUB 1300
1100 CALL HCHAR(R,C(I,1),RC(I,2),136)
1110 NEXT I
1120 CALL HCHAR(12,16,96)
1130 CALL SCREEN(PL)
1140 CALL HCHAR(5,4,32,3)
1150 A=5
1160 X=4
1170 GOSUB 1280
1180 O$=S$
1190 GOSUB 1300
1200 X=5
1210 CALL HCHAR(5,23,32,3)
1220 IF S$(P)=S$(PL)THEN 2390
1230 IF S$(P)=S$(PL)THEN 1860
1240 IF S$(P)=S$(PL)THEN 2370
1250 CALL HCHAR(24,14,32,2)
1260 CALL HCHAR(24,20,32,2)
1270 A=0
1280 X=24
1290 GOSUB 1310
1300 X=1
1310 O=O+1
1320 CALL SOUND(-250,S(O),0)
1330 IF O<3 THEN 1320
1340 IF P=1 THEN 1070
1350 IF P=2 THEN 1400
1360 IF P=3 THEN 2100
1370 RESTORE 2860
1380 CALL HCHAR(24,14,32,2)
1390 GOTO 1070
1400 O=20
1410 GOSUB 1310
1420 IF P=1 THEN 1470
1430 IF P=2 THEN 2100
1440 IF P=3 THEN 2100
1450 FOR I=1 TO 30
1460 IF S(I)>33 THEN 2100
1470 NEXT I
1480 CALL HCHAR(R,C,54)
1490 CALL HCHAR(R,C,32)
1500 R=R+SGN(R-MR)*(ABS(R-MR)>1)
1510 C=C+SGN(C-MC)*(ABS(C-MC)>1)
1520 CALL HCHAR(MR,MC,32)
1530 CALL HCHAR(R,C,30)
1540 CALL HCHAR(R,C,32)
1550 CALL HCHAR(R,C,30)
1560 CALL HCHAR(R,C,32)
1570 MR=R
1580 MC=C
1590 CALL KEY(0,K,S)
1600 IF S=0 THEN 1590
1610 IF K=83 THEN C=C-1
1620 IF K=68 THEN C=C+1
1630 IF K=69 THEN R=R-1
1640 IF K=88 THEN R=R+1
1650 IF K=32 THEN 1670
1660 GOTO 1590
1670 CALL HCHAR(R,C,30)
1680 CALL GCHAR(R,C,V)
1690 IF V=32 THEN 1590
1700 CALL HCHAR(R,C,32)
1710 CALL HCHAR(MR,MC,30)
1720 CALL HCHAR(MR,MC,32)
1730 CALL HCHAR(R,C,30)
1740 CALL HCHAR(R,C,32)
1750 CALL HCHAR(R,C,30)
1760 CALL HCHAR(R,C,32)
1770 CALL HCHAR(R,C,30)
1780 CALL HCHAR(R,C,32)
1790 CALL HCHAR(R,C,30)
1800 CALL HCHAR(R,C,32)
1810 CALL HCHAR(R,C,30)
1820 CALL HCHAR(R,C,32)
1830 CALL HCHAR(R,C,30)
1840 CALL HCHAR(R,C,32)
1850 CALL HCHAR(R,C,30)
1860 CALL HCHAR(R,C,32)
1870 CALL HCHAR(R,C,30)
1880 CALL HCHAR(R,C,32)
1890 CALL HCHAR(R,C,30)
1900 CALL HCHAR(R,C,32)
1910 CALL HCHAR(R,C,30)
1920 CALL HCHAR(R,C,32)
1930 CALL HCHAR(R,C,30)
1940 CALL HCHAR(R,C,32)
1950 CALL HCHAR(R,C,30)
1960 CALL HCHAR(R,C,32)
1970 CALL HCHAR(R,C,30)
1980 CALL HCHAR(R,C,32)
1990 CALL HCHAR(R,C,30)
2000 CALL HCHAR(R,C,32)
2010 CALL HCHAR(R,C,30)
2020 CALL HCHAR(R,C,32)
2030 CALL HCHAR(R,C,30)
2040 CALL HCHAR(R,C,32)
2050 CALL HCHAR(R,C,30)
2060 CALL HCHAR(R,C,32)
2070 CALL HCHAR(R,C,30)
2080 CALL HCHAR(R,C,32)
2090 CALL HCHAR(R,C,30)
2100 CALL HCHAR(R,C,32)
2110 CALL HCHAR(R,C,30)
2120 CALL HCHAR(R,C,32)
2130 CALL HCHAR(R,C,30)
2140 CALL HCHAR(R,C,32)
2150 CALL HCHAR(R,C,30)
2160 CALL HCHAR(R,C,32)
2170 CALL HCHAR(R,C,30)
2180 CALL HCHAR(R,C,32)
2190 CALL HCHAR(R,C,30)
2200 CALL HCHAR(R,C,32)
2210 CALL HCHAR(R,C,30)
2220 CALL HCHAR(R,C,32)
2230 CALL HCHAR(R,C,30)
2240 CALL HCHAR(R,C,32)
2250 CALL HCHAR(R,C,30)
2260 CALL HCHAR(R,C,32)
2270 CALL HCHAR(R,C,30)
2280 CALL HCHAR(R,C,32)
2290 CALL HCHAR(R,C,30)
2300 CALL HCHAR(R,C,32)
2310 CALL HCHAR(R,C,30)
2320 CALL HCHAR(R,C,32)
2330 CALL HCHAR(R,C,30)
2340 CALL HCHAR(R,C,32)
2350 CALL HCHAR(R,C,30)
2360 CALL HCHAR(R,C,32)
2370 CALL HCHAR(R,C,30)
2380 CALL HCHAR(R,C,32)
2390 CALL HCHAR(R,C,30)
2400 CALL HCHAR(R,C,32)
2410 CALL HCHAR(R,C,30)
2420 CALL HCHAR(R,C,32)
2430 CALL HCHAR(R,C,30)
2440 CALL HCHAR(R,C,32)
2450 CALL HCHAR(R,C,30)
2460 CALL HCHAR(R,C,32)
2470 CALL HCHAR(R,C,30)
2480 CALL HCHAR(R,C,32)
2490 CALL HCHAR(R,C,30)
2500 CALL HCHAR(R,C,32)
2510 CALL HCHAR(R,C,30)
2520 CALL HCHAR(R,C,32)
2530 CALL HCHAR(R,C,30)
2540 CALL HCHAR(R,C,32)
2550 CALL HCHAR(R,C,30)
2560 CALL HCHAR(R,C,32)
2570 CALL HCHAR(R,C,30)
2580 CALL HCHAR(R,C,32)
2590 CALL HCHAR(R,C,30)
2600 CALL HCHAR(R,C,32)
2610 CALL HCHAR(R,C,30)
2620 CALL HCHAR(R,C,32)
2630 CALL HCHAR(R,C,30)
2640 CALL HCHAR(R,C,32)
2650 CALL HCHAR(R,C,30)
2660 CALL HCHAR(R,C,32)
2670 CALL HCHAR(R,C,30)
2680 CALL HCHAR(R,C,32)
2690 CALL HCHAR(R,C,30)
2700 CALL HCHAR(R,C,32)
2710 CALL HCHAR(R,C,30)
2720 CALL HCHAR(R,C,32)
2730 CALL HCHAR(R,C,30)
2740 CALL HCHAR(R,C,32)
2750 CALL HCHAR(R,C,30)
2760 CALL HCHAR(R,C,32)
2770 CALL HCHAR(R,C,30)
2780 CALL HCHAR(R,C,32)
2790 CALL HCHAR(R,C,30)
2800 CALL HCHAR(R,C,32)
2810 CALL HCHAR(R,C,30)
2820 CALL HCHAR(R,C,32)
2830 CALL HCHAR(R,C,30)
2840 CALL HCHAR(R,C,32)
2850 DATA 1,1,2,1,3,1,4,1,5,1
2860 DATA 1,2,2,2,3,2,4,2,5,2
2870 CALL HCHAR(24,14,32,2)
2880 CALL HCHAR(24,20,32,2)
2890 A=0
2900 X=24
2910 GOSUB 1310
2920 X=1
2930 O=O+1
2940 CALL SOUND(-250,S(O),0)
2950 IF O<3 THEN 2940
2960 IF P=1 THEN 1070
2970 IF P=2 THEN 1400
2980 IF P=3 THEN 2100
2990 RESTORE 2860
3000 CALL HCHAR(24,14,32,2)
3010 GOTO 1070
```

PEG JUMP *Continued*

TI-99/4A

```
2790 DATA 0000000000E72144781
2800 DATA 0000000000E72254781
2810 DATA 0000000000E72254781
2820 DATA 0000000000E72261E21
2830 DATA 1,192,94,8,95,54,94,21,17,95,21,2,2
2840 DATA 1,27,7,11,11,PEGS,4,4,LEFT,3,23,PEGS,4,
2850 DATA 23,REMOVE,15,26,31,18,3,11,16,25,12,18,6,27
2860 DATA 24,24,6,31,18,3,10,12,11,10,12
2870 DATA 13,11,16,12,18,3,10,12,11,10,12
```

HCM

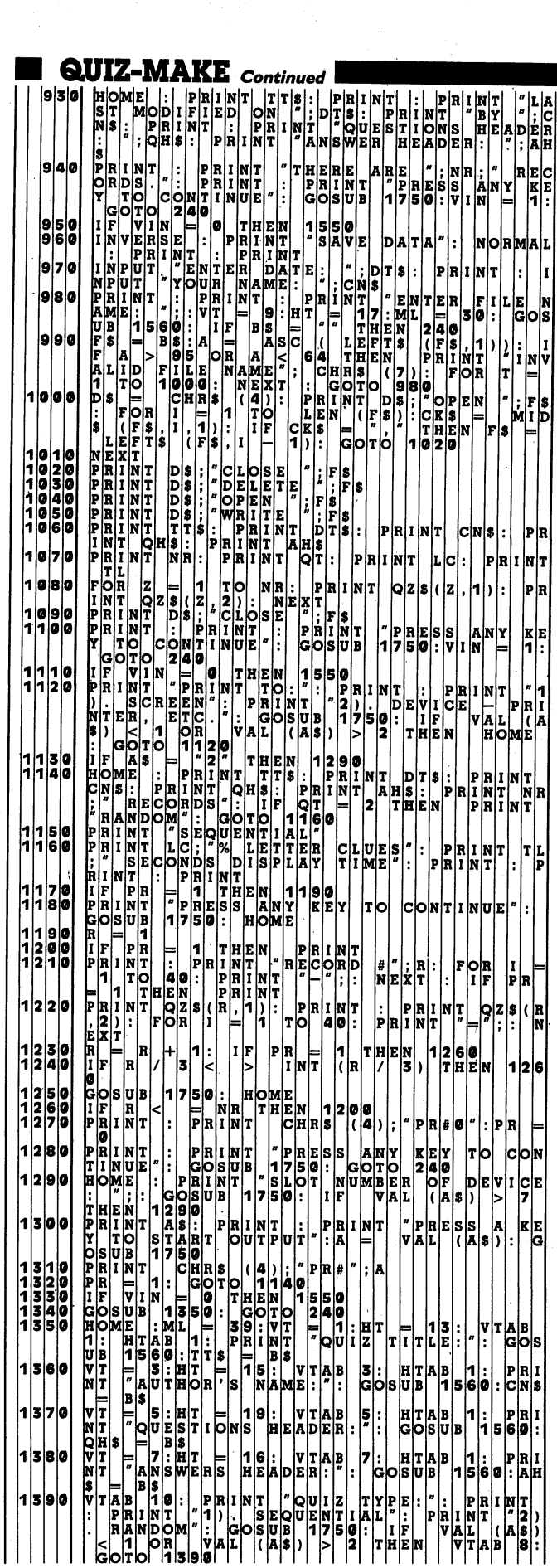
QUIZ-MAKE

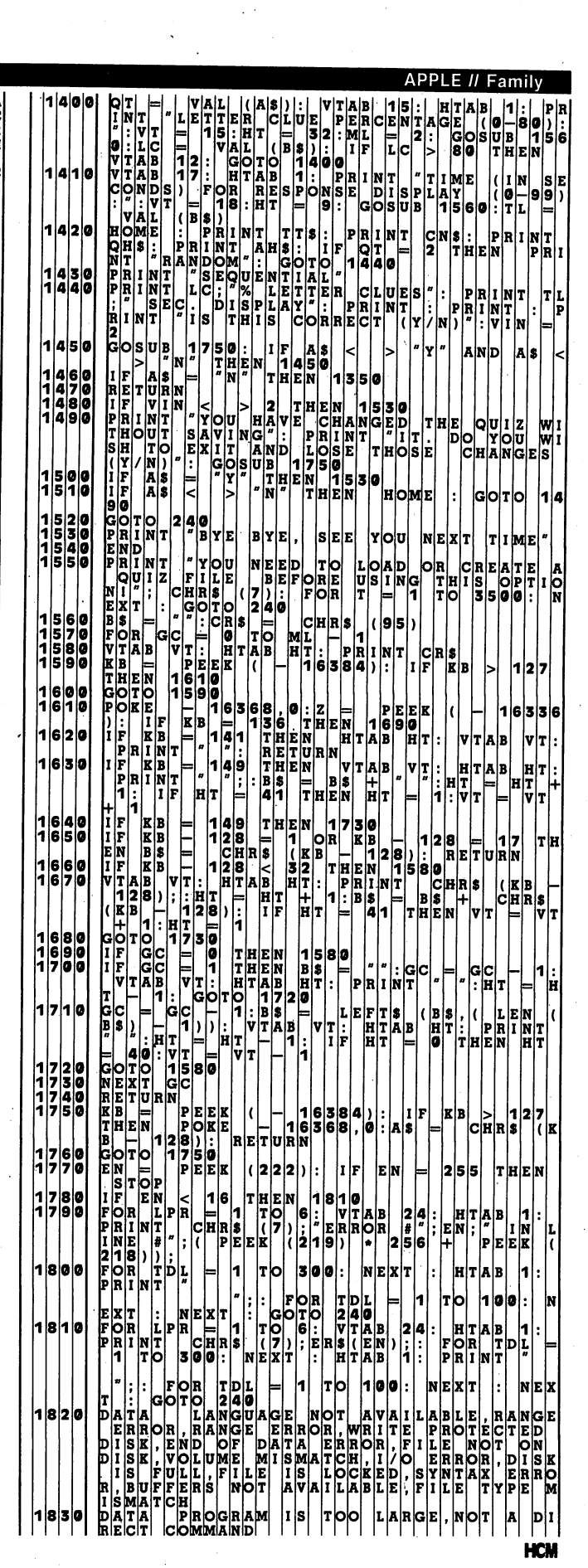
APPLE // Family

```
100 REM MAKE KEYSTROKE
110 REM AMCMPUTER.SOFT
120 REM BY ANDREW VAP
130 REM
140 REM
150 REM
160 REM
170 REM
180 REM
190 REM
200 HOME : VTAB 24: PRINT "QUIZ-MAKER 1.0"
210 INPUT "FILE NAME TO SAVE QUIZ ON? ";FILE$
220 PRINT CHR$(4);"OPEN";FILE$
230 PRINT CHR$(4);"DELETE";FILE$
240 PRINT CHR$(4);"OPEN";FILE$
250 INPUT "HOW MANY QUESTIONS? ";NUMQUES
260 PRINT CHR$(4);"WRITE";FILE$
270 PRINT NUMQUES
280 FOR I = 1 TO NUMQUES
290 PRINT "QUESTION #";I;":"
300 INPUT Q$
310 PRINT "ANSWER #";I;":"
320 INPUT A$
330 PRINT Q$
340 PRINT A$
350 NEXT I
360 PRINT CHR$(4);"CLOSE"
370 END
380 REM ** ERROR TRAP **
390 PRINT CHR$(4);"CLOSE"
400 PRINT "ERROR #";PEEK(222)
410 PRINT "LINE #";PEEK(218) + PEEK(219) * 256
420 END
430 REM ** ERROR TRAP **
440 PRINT CHR$(4);"CLOSE"
450 PRINT "ERROR #";PEEK(222)
460 PRINT "LINE #";PEEK(218) + PEEK(219) * 256
470 END
480 REM ** ERROR TRAP **
490 PRINT CHR$(4);"CLOSE"
500 PRINT "ERROR #";PEEK(222)
510 PRINT "LINE #";PEEK(218) + PEEK(219) * 256
520 END
530 REM ** ERROR TRAP **
540 PRINT CHR$(4);"CLOSE"
550 PRINT "ERROR #";PEEK(222)
560 PRINT "LINE #";PEEK(218) + PEEK(219) * 256
570 END
580 REM ** ERROR TRAP **
590 PRINT CHR$(4);"CLOSE"
600 PRINT "ERROR #";PEEK(222)
610 PRINT "LINE #";PEEK(218) + PEEK(219) * 256
620 END
630 REM ** ERROR TRAP **
640 PRINT CHR$(4);"CLOSE"
650 PRINT "ERROR #";PEEK(222)
660 PRINT "LINE #";PEEK(218) + PEEK(219) * 256
670 END
680 REM ** ERROR TRAP **
690 PRINT CHR$(4);"CLOSE"
700 PRINT "ERROR #";PEEK(222)
710 PRINT "LINE #";PEEK(218) + PEEK(219) * 256
720 END
730 REM ** ERROR TRAP **
740 PRINT CHR$(4);"CLOSE"
750 PRINT "ERROR #";PEEK(222)
760 PRINT "LINE #";PEEK(218) + PEEK(219) * 256
770 END
780 REM ** ERROR TRAP **
790 PRINT CHR$(4);"CLOSE"
800 PRINT "ERROR #";PEEK(222)
810 PRINT "LINE #";PEEK(218) + PEEK(219) * 256
820 END
830 REM ** ERROR TRAP **
840 PRINT CHR$(4);"CLOSE"
850 PRINT "ERROR #";PEEK(222)
860 PRINT "LINE #";PEEK(218) + PEEK(219) * 256
870 END
880 REM ** ERROR TRAP **
890 PRINT CHR$(4);"CLOSE"
900 PRINT "ERROR #";PEEK(222)
910 PRINT "LINE #";PEEK(218) + PEEK(219) * 256
920 END
```

Continued

PROGRAM LISTING





QUIZ-MAKE


```
100 REM *************
110 REM * QUIZ-MAKE *
120 REM *************
130 REM BY WILLIAM K. BALTHROP
140 REM HOME COMPUTER MAGAZINE
150 REM VERSION 4.5.1
160 REM C-64 BASIC
170 REM
180 REM INITIALIZE THE PROGRAM, AND DISPLAY THE TITLE
190 PRINT"{SHIFT CLR}"
200 MX=40:DIM QZ$(MX,2):POKE 650,128
210 POKE 53280,6:POKE 53281,12:POKE 646,0:POKE 53272,21:POKE 657,128
220 X=7:Y=13:GOSUB3050:PRINT"** QUIZ-MAKE **"
230 X=8:Y=24:GOSUB3050:PRINT"{CTRL RVSON} PRESS RETURN TO START ";
240 GOSUB2700
250 IF ASC(K$)<>13 THEN240
260 REM MAIN MENU
270 POKE 198,0
280 PRINT"{SHIFT CLR}":PRINTTAB(16)"QUIZ-MAKE"
290 X=0:Y=6:GOSUB3050
300 PRINT" 1) EDIT":PRINT
310 PRINT" 2) LOAD":PRINT:PRINT" 3) SAVE":PRINT:PRINT" 4) PRINT":PRINT
320 PRINT" 5) CHANGE PARAMETERS":PRINT:PRINT" 6) EXIT"
330 X=4:Y=24:GOSUB3050:PRINT"{CTRL RVSON} INPUT A NUMBER BETWEEN 1 AND 6 ";
340 GOSUB2700
350 IF ASC(K$)<49 OR ASC(K$)>54 THEN340
360 ON VAL(K$) GOSUB380,1140,1460,1790,2210,3070
370 GOTO260
380 REM EDIT THE QUIZ
390 PRINT"{SHIFT CLR}"
400 IF NR<MX THEN450
410 X=10:Y=10:GOSUB3050
420 PRINT"RECORDS ARE FULL!!":PRINT:PRINTTAB(16)"SORRY"
430 FOR I=1 TO 2300:NEXT
440 GOTO750
450 IF VIN<>0 THEN470
460 NR=0:GOSUB2230:GOTO380
470 WR=NR+1
480 PRINTTAB(14)"EDIT MODE"
490 PRINTTAB((39-LEN(TL$))/2)TL$
500 PRINTTAB(13)"RECORD # "WR
510 PRINT:PRINT:PRINTQT$
520 FOR I=1 TO 40:PRINT"{SHIFT *}";:NEXT
530 PRINT:PRINT
540 FOR I=1 TO 40:PRINT"{SHIFT *}";:NEXT
550 PRINT:PRINT:PRINT:PRINTAT$
560 FOR I=1 TO 40:PRINT"{SHIFT *}";:NEXT
570 PRINT
580 FOR I=1 TO 40:PRINT"{SHIFT *}";:NEXT:PRINT:PRINT"{5 CRSR DOWN}PRESS [RETURN] TO RETURN TO MENU";
590 X=13:Y=3:GOSUB3050
600 PRINT"RECORD # "WR"{SHIFT CRSR LEFT}"
610 LN=8:GOSUB3030:LN=9:GOSUB3030
620 LN=16:GOSUB3030
630 Y=8:X=0:GOSUB3050
640 B=WR:QZ$(WR,1)="":GOSUB2730
650 IF ASC(K$)=209 THEN QA=1:LN=8:GOSUB3030:LN=9:GOSUB3030:GOTO760
660 IF ASC(K$)=193 THEN QA=2:LN=16:GOSUB3030:GOTO760
670 IF L=0 THEN QZ$(WR,1)="":QZ$(WR,2)="":GOTO750
680 Y=16:X=0:GOSUB3050
690 B=WR:QZ$(WR,2)="":GOSUB2880
700 IF ASC(K$)=209 THEN QA=1:LN=8:GOSUB3030:LN=9:GOSUB3030:GOTO760
710 IF ASC(K$)=193 THEN QA=2:LN=16:GOSUB3030:GOTO760
720 IF L=0 THEN QZ$(WR,1)="":QZ$(WR,2)="":GOTO750
730 VIN=2:WR=WR+1:IF WR>MX THEN PRINT"{SHIFT CLR}":GOTO410
740 GOTO590
750 NR=WR-1:RETURN
760 REM SEARCH MODE
770 Y=5:IF QA=2 THEN Y=13
780 X=0:GOSUB3050:PRINT"SEARCH FOR";
790 LN=8:GOSUB3030:LN=9:GOSUB3030:LN=16:GOSUB3030:Y=QA*8:GOSUB3050
800 LL=80/QA:KL=32:KU=90:GOSUB2520:S$=ST$
810 IF S$="" THEN920
820 QZ$(WR,1)="":QZ$(WR,2)="":S=1
830 IF WR=S THEN920
840 LS=LEN(S$):F=0
850 LN=LEN(QZ$(S,QA))-LS+1
860 IF LN<1 THEN900
870 FOR I=1 TO LN
880 IF MID$(QZ$(S,QA),I,LS)=S$ THEN F=S
890 NEXT:IF F<>0 THEN910
900 S=S+1:IF S<>WR THEN850
910 IF F<>0 THEN950
920 X=13:Y=19:GOSUB3050:PRINT" NOT FOUND"
930 FOR I=1 TO 2000:NEXT:LN=19:GOSUB3030:LN=5:GOSUB3030:LN=13:GOSUB3030
940 LN=16:GOSUB3030:GOTO590
950 X=13:Y=3:GOSUB3050:PRINT"RECORD # "F"{SHIFT CRSR LEFT}"
960 LN=8:GOSUB3030:LN=9:GOSUB3030:X=0:Y=8:GOSUB3050:PRINT QZ$(F,1)
970 LN=16:GOSUB3030:X=0:Y=16:GOSUB3050:PRINT QZ$(F,2)
980 X=4:Y=19:GOSUB3050:PRINT"[C]-{CTRL RVSON}C{CTRL RVSOFF}HANGE [N]-{CTRL RVSON}N{CTRL RVSOFF}EXT [E]-{CTRL RVSON}E{CTRL RVSOFF}XIT"
990 GOSUB2700:IF K$<>"C" AND K$<>"N" AND K$<>"E" THEN990
1000 LN=19:GOSUB3030
1010 IF K$<>"C" THEN1120
1020 VIN=2:IF QA=2 THEN1080
1030 LN=8:GOSUB3030:LN=9:GOSUB3030:X=0:Y=8:GOSUB3050
1040 QZ$(F,1)="":B=F:GOSUB2730
1050 IF ASC(K$)=209 OR ASC(K$)=193 THEN GOSUB2750:GOTO1050
1060 IF L=0 THEN1030
1070 GOTO1100
1080 LN=16:GOSUB3030:X=0:Y=16:GOSUB3050:QZ$(F,2)="":B=F:GOSUB2880
1090 IF ASC(K$)=209 OR ASC(K$)=193 THEN GOSUB2900:GOTO1090
1100 IF L=0 THEN1080
1110 GOTO1130
1120 IF K$="N" THEN S=S+1:GOTO830
1130 LN=5:GOSUB3030:LN=19:GOSUB3030:LN=13:GOSUB3030:GOTO590
1140 REM LOAD QUIZ FILE
1150 PRINT"{SHIFT CLR}":VIN=1
1160 PRINTTAB(14)"LOAD DATA":PRINT:PRINT:PRINT:PRINT
1170 PRINT"FILE NAME :";:LL=14:KL=65:KU=90:GOSUB2520:PRINT
1180 X$=ST$
1190 PRINT"TAPE OR DISK (T/D) ";
1200 GOSUB2700:IF K$<>"T" AND K$<>"D" THEN1200
1210 DV$="{CTRL RVSON}TAPE":IF K$="D" THEN DV$="{CTRL RVSON}DISK"
1220 PRINTDV$:PRINT
1230 IF DV$="{CTRL RVSON}TAPE" THEN1250
1240 OPEN 1,8,8,"0:"+X$+".Q"+",S,R":GOSUB1260:RETURN
1250 OPEN 1,1,0,X$+".Q":GOSUB1260:RETURN
1260 INPUT#1,TL$
1270 INPUT#1,DT$
1280 INPUT#1,CN$
1290 INPUT#1,QT$
1300 INPUT#1,AT$
1310 INPUT#1,NR
1320 INPUT#1,QT
1330 INPUT#1,LCP
1340 INPUT#1,TD:IF TL$="" THEN VIN=0:GOTO 1410
1350 PRINTTL$" LAST MODIFIED ON "DT$:PRINT"BY "CN$:PRINT
1360 PRINT"QUESTIONS HEADER :"QT$:PRINT"ANSWERS HEADER :"AT$:PRINT
1370 PRINT"THERE ARE"NR" RECORDS":IF NR=0 THEN1410
1380 FOR I=1 TO NR
1390 INPUT#1,QZ$(I,1)
1400 INPUT#1,QZ$(I,2):NEXT
1410 IF DV$="{CTRL RVSON}TAPE" THEN 1430
1420 OPEN15,8,15:INPUT#15,V,S$:CLOSE 15:IF V<>0 THEN PRINT:PRINTS$
1430 X=5:Y=24:GOSUB3050:PRINT"{CTRL RVSON} PRESS RETURN TO CONTINUE ";
1440 GOSUB2700:IF ASC(K$)<>13 THEN1440
1450 CLOSE1:RETURN
1460 REM SAVE QUIZ FILE
1470 IF VIN=0 THEN GOSUB2640:RETURN
1480 PRINT"{SHIFT CLR}":H=VIN:VIN=1
1490 PRINTTAB(14)"SAVE DATA":PRINT
1500 PRINT"ENTER DATE :";:LL=25:KL=32:KU=90:GOSUB2520:PRINT
1510 DT$=ST$
1520 PRINT"ENTER YOUR NAME :";:GOSUB2520:PRINT
1530 CN$=ST$
1540 PRINT"FILE NAME :";:LL=14:KL=65:GOSUB2520:PRINT
1550 X$=ST$
1560 PRINT"TAPE OR DISK (T/D) ";
1570 GOSUB2700:IF K$<>"T" AND K$<>"D" THEN1570
1580 DV$="{CTRL RVSON}TAPE":IF K$="D" THEN DV$="{CTRL RVSON}DISK"
1590 PRINTDV$
1600 IF DV$="{CTRL RVSON}TAPE" THEN1620
```

Continued

PROGRAM LISTING

```
1610 OPEN 1,8,8,"@0:"+X$+".Q"+",S,W":GOSUB1630:RETURN
1620 OPEN 1,1,1,X$+".Q":GOSUB1630:RETURN
1630 PRINT#1,TL$
1640 PRINT#1,DT$
1650 PRINT#1,CN$
1660 PRINT#1,QT$
1670 PRINT#1,AT$
1680 PRINT#1,NR
1690 PRINT#1,QT
1700 PRINT#1,LCP
1710 PRINT#1,TD
1720 FOR I=1 TO NR
1730 PRINT#1,QZ$(I,1)
1740 PRINT#1,QZ$(I,2):NEXT:IF DV$="{CTRL RVSON}TAPE" THEN 1760
1750 OPEN15,8,15:INPUT#15,V,S$:CLOSE15:IF V<>0 THEN PRINT:PRINT S$:VIN=H
1760 X=5:Y=24:GOSUB3050:PRINT"{CTRL RVSON} PRESS RETURN TO CONTINUE ";
1770 GOSUB2700:IF ASC(K$)<>13 THEN1770
1780 CLOSE1:RETURN
1790 REM PRINT QUIZ FILE
1800 IF VIN=0 THEN GOSUB2640:RETURN
1810 PRINT"{SHIFT CLR}":POKE 198,0
1820 PRINT"PRINT TO 1) SCREEN 2) PRINTER"
1830 GOSUB2700:IF K$<>"1" AND K$<>"2" THEN1830
1840 PRINT"{SHIFT CLR}"
1850 IF K$="2" THEN2030
1860 PRINT:PRINTTL$:PRINTDT$:PRINTCN$:PRINTQT$:PRINTAT$:PRINTNR "RECORDS"
1870 IF QT=2 THEN1890
1880 PRINT"SEQUENTIAL":GOTO1900
1890 PRINT"RANDOM"
1900 PRINTLCP"% LETTER CLUES":PRINTTD" SECONDS DISPLAY TIME"
1910 X=5:Y=24:GOSUB3050:PRINT"{CTRL RVSON} PRESS RETURN TO CONTINUE ";
1920 GOSUB2700:IF ASC(K$)<>13 THEN1920
1930 R=1:IF NR=0 THEN2020
1940 PRINT"{SHIFT CLR}":PRINT:PRINT:PRINT
1950 PRINT"RECORD #"R"{SHIFT CRSRLEFT}":PRINTQZ$(R,1):PRINTQZ$(R,2):PRINT
1960 R=R+1:IF R>NR THEN1990
1970 IF R/4 <> INT(R/4) THEN1950
1980 PRINT"RECORD #"R"{SHIFT CRSRLEFT}":PRINTQZ$(R,1):PRINTQZ$(R,2):PRINT
1990 X=5:Y=24:GOSUB3050:PRINT"{CTRL RVSON} PRESS RETURN TO CONTINUE ";
2000 GOSUB2700:IF ASC(K$)<>13 THEN2000
2010 IF R<NR THEN R=R+1:GOTO1940
2020 RETURN
2030 X=15:Y=10:GOSUB3050:PRINT"PRINTING"
2040 OPEN 4,4:CMD4
2050 PRINTTL$:PRINTDT$:PRINTCN$:PRINTQT$:PRINTAT$:PRINTNR" RECORDS"
2060 IF QT=2 THEN2080
2070 PRINT"SEQUENTIAL":GOTO2090
2080 PRINT"RANDOM"
2090 PRINTLCP"% LETTER CLUES":PRINTTD" SECONDS DISPLAY TIME"
2100 PRINT:PRINT:PRINT:IF NR=0 THEN2170
2110 FOR I=1 TO NR
2120 PRINT"RECORD #"I" {SHIFT CRSRLEFT}"
2130 PRINTQZ$(I,1)
2140 PRINTQZ$(I,2)
2150 PRINT
2160 NEXT
2170 PRINT#4:CLOSE 4,4
2180 X=5:Y=24:GOSUB3050:PRINT"{CTRL RVSON} PRESS RETURN TO CONTINUE ";
2190 GOSUB2700:IF ASC(K$)<>13 THEN2190
2200 RETURN
2210 REM CHANGE PARAMETERS ROUTINE
2220 IF VIN=0 THEN GOSUB2640:RETURN
2230 PRINT"{SHIFT CLR}"
2240 PRINT"QUIZ TITLE :";:LL=16:KL=32:KU=90:GOSUB2520:PRINT
2250 TL$=ST$
2260 PRINT"AUTHOR'S NAME :";:LL=25:GOSUB2520:PRINT
2270 CN$=ST$
2280 PRINT"QUESTIONS HEADER :";:GOSUB2520:PRINT
2290 QT$=ST$
2300 PRINT"ANSWERS HEADER :";:GOSUB2520:PRINT
2310 AT$=ST$
2320 PRINT"QUIZ TYPE; 1) SEQUENTIAL 2) RANDOM :";
2330 LL=1:KL=49:KU=50:GOSUB2520:PRINT
2340 QT=VAL(ST$)
2350 PRINT"LETTER CLUE PERCENTAGE (0-80) :";
2360 LL=2:KL=48:KU=57:GOSUB2520:PRINT
2370 LCP=VAL(ST$):IF LCP>80 THEN2350
2380 PRINT"TIME (IN SECONDS) FOR":PRINT"RESPONSE DISPLAY (0-99) :";
2390 LL=2:KU=57:GOSUB2520
2400 TD=VAL(ST$)
2410 PRINT"{SHIFT CLR}"
2420 PRINTTL$:PRINTCN$:PRINTQT$:PRINTAT$
2430 IF QT=2 THEN2450
2440 PRINT"SEQUENTIAL":GOTO2460
2450 PRINT"RANDOM"
2460 PRINTLCP"% LETTER CLUES":PRINTTD" SEC. DISPLAY"
2470 FOR I=1 TO 40:PRINT"{SHIFT *}";:NEXT
2480 PRINT"IS THIS CORRECT (Y/N)"
2490 GOSUB2700:IF K$<>"Y" AND K$<>"N" THEN2490
2500 VIN=2:IF K$="N" THEN2210
2510 RETURN
2520 REM INPUT ROUTINE
2530 L=0:ST$=""
2540 PRINT"{CMDR P}{SHIFT CRSRLEFT}";:GOSUB2700
2550 IF ASC(K$)=13 AND L<>0 THEN PRINT" ":GOTO2630
2560 IF ASC(K$)<>20 OR L=0 THEN2580
2570 ST$=LEFT$(ST$,LEN(ST$)-1):PRINT" {2 SHIFT CRSRLEFT}";:L=L-1:GOTO2540
2580 IF ASC(K$)<KL OR ASC(K$)>KU THEN2540
2590 IF ASC(K$)=34 THEN2540
2600 PRINTK$;:ST$=ST$+K$:L=L+1
2610 IF L=LL THEN2630
2620 GOTO2540
2630 RETURN
2640 REM ILLEGAL ENTRY MESSAGE
2650 PRINT"{SHIFT CLR}"
2660 X=6:Y=9:GOSUB3050:PRINT"YOU NEED TO LOAD OR CREATE":PRINT
2670 PRINTTAB(7)"A QUIZ FILE BEFORE USING":PRINT
2680 PRINTTAB(10)"USING THIS OPTION"
2690 FOR I=1 TO 3250:NEXT:RETURN
2700 REM SINGLE KEY INPUT ROUTINE
2710 GET K$:IF K$="" THEN2710
2720 POKE 198,0:RETURN
2730 REM ROUTINE TO INPUT A QUESTION
2740 L=0
2750 PRINT"{CMDR P}{SHIFT CRSRLEFT}";:GOSUB2700
2760 IF ASC(K$)=13 THEN PRINT" ":GOTO2870
2770 IF ASC(K$)<>20 OR L=0 THEN2800
2780 QZ$(B,1)=LEFT$(QZ$(B,1),LEN(QZ$(B,1))-1)
2790 L=L-1:PRINT" {2 SHIFT CRSRLEFT}";:GOTO2750
2800 IF ASC(K$)=209 OR ASC(K$)=193 THEN2870
2810 IF ASC(K$)<32 OR ASC(K$)>90 THEN2750
2820 IF ASC(K$)=34 THEN2750
2830 L=L+1
2840 PRINTK$;:QZ$(B,1)=QZ$(B,1)+K$
2850 IF L=80 THEN2870
2860 GOTO2750
2870 RETURN
2880 REM ROUTINE TO INPUT A ANSWER
2890 L=0
2900 PRINT"{CMDR P}{SHIFT CRSRLEFT}";:GOSUB2700
2910 IF ASC(K$)=13 THEN PRINT" ":GOTO3020
2920 IF ASC(K$)<>20 OR L=0 THEN2950
2930 QZ$(B,2)=LEFT$(QZ$(B,2),LEN(QZ$(B,2))-1)
2940 L=L-1:PRINT" {2 SHIFT CRSRLEFT}";:GOTO2900
2950 IF ASC(K$)=209 OR ASC(K$)=193 THEN3020
2960 IF ASC(K$)<32 OR ASC(K$)>90 THEN2900
2970 IF ASC(K$)=34 THEN2900
2980 L=L+1
2990 PRINTK$;:QZ$(B,2)=QZ$(B,2)+K$
3000 IF L=39 THEN3020
3010 GOTO2900
3020 RETURN
3030 REM CLEAR PARTS OF THE EDIT SCREEN
3040 POKE 781,LN:SYS 59903:RETURN
3050 REM ROUTINE TO SIMULATE THE PRINT AT STATEMENT
3060 POKE 781,Y:POKE 782,X:POKE 783,0:SYS 65520:RETURN
3070 REM EXIT PROGRAM ROUTINE
3080 PRINT"{SHIFT CLR}":PRINT:PRINT:PRINT
3090 IF VIN<>2 THEN3160
3100 PRINT"YOU HAVE CHANGED THE QUIZ"
3110 PRINT"WITHOUT SAVING IT.":PRINT
3120 PRINT"DO YOU WISH TO EXIT AND"
3130 PRINT"LOSE THOSE CHANGES (Y/N)"
3140 GOSUB2700:IF K$<>"Y" AND K$<>"N" THEN3140
3150 IF K$="N" THEN RETURN
3160 PRINT"{SHIFT CLR}":PRINT"BYE BYE, SEE YOU NEXT TIME..."
3170 POKE 53280,14:POKE 53281,6:POKE 646,14:POKE 650,0:POKE 657,0:END
```

HCM

QUIZ-MAKE

```
100 '
110 '  ********************************
120 '  *                              *
130 '  *          QUIZ-MAKE           *
140 '  *                              *
150 '  *  BY BRUCE M. BOMBERGER JR.   *
160 '  *                              *
170 '  *  COPYRIGHT (C) 1984 BY IBM   *
180 '  *                              *
190 '  *  ALL RIGHTS RESERVED WORLD-  *
200 '  *  WIDE.                       *
210 '  *                              *
220 '  ********************************
230 '
240 '
250 '
260 '
270 '
280 '
290 '
300 '
310 '
320 '
330 '
340 '
350 '
360 '
370 '
380 '
390 '
400 '
410 '
420 '
430 '
440 '
450 '
460 '
470 '
480 '
490 '
500 '
510 '
520 '
530 '
540 '
550 '
560 '
570 '
580 '
590 '
600 '
610 '
620 '
630 '
640 '
650 '
660 '
670 '
680 '
690 '
700 '
710 '
720 '
730 '
740 '
750 '
760 '
770 '
780 '
790 '
800 '
810 '
820 '
```

Continued

QUIZ-MAKE *Continued*

IBM PC & IBM PCjr

```
830 LOCATE 20,1:PRINT "SAVING RECORD #";Z$:PRINT:PRINT "ENTER #1,QZ$(Z,1),QZ$(Z,2),QZ$(Z,3)"
840 FOR Z=1 TO NR:WRITE #1,QZ$(Z,1),QZ$(Z,2),QZ$(Z,3):NEXT Z:CLOSE #1:PRINT "SAVED":END
850 CLS:LOCATE 22,1:CLEAR 1000:DIM QZ$(NR,3):OPEN "I",#1,"QUIZ.DAT":FOR Z=1 TO NR:INPUT #1,QZ$(Z,1),QZ$(Z,2),QZ$(Z,3):NEXT Z:CLOSE #1
860 CLS:PRINT "QUIZ FILE CONTENTS":PRINT:FOR Z=1 TO NR:PRINT Z;QZ$(Z,1):NEXT Z:PRINT:INPUT "ENTER RECORD # TO CORRECT";Z:PRINT:PRINT "ENTER #1,QZ$(Z,1),QZ$(Z,2),QZ$(Z,3)"
870 INPUT #1,QZ$(Z,1),QZ$(Z,2),QZ$(Z,3):PRINT:PRINT "SAVING RECORD #";Z$:PRINT:PRINT "ENTER #1,QZ$(Z,1),QZ$(Z,2),QZ$(Z,3)"
880 GOTO 870
890 CLS:PRINT "QUIZ FILE CONTENTS":PRINT:FOR Z=1 TO NR:PRINT Z;QZ$(Z,1):NEXT Z:PRINT:INPUT "ENTER RECORD # TO DELETE";Z:PRINT:PRINT "ENTER #1,QZ$(Z,1),QZ$(Z,2),QZ$(Z,3)"
900 INPUT #1,QZ$(Z,1),QZ$(Z,2),QZ$(Z,3):PRINT:PRINT "SAVING RECORD #";Z$:PRINT:PRINT "ENTER #1,QZ$(Z,1),QZ$(Z,2),QZ$(Z,3)"
910 GOTO 900
920 CLS:LOCATE 20,1:PRINT "ENTER RECORD # TO BE INSERTED":INPUT Z:PRINT:PRINT "ENTER #1,QZ$(Z,1),QZ$(Z,2),QZ$(Z,3)"
930 INPUT #1,QZ$(Z,1),QZ$(Z,2),QZ$(Z,3):PRINT:PRINT "SAVING RECORD #";Z$:PRINT:PRINT "ENTER #1,QZ$(Z,1),QZ$(Z,2),QZ$(Z,3)"
940 GOTO 930
950 CLS:PRINT "ENTER #1,QZ$(1,1),QZ$(1,2),QZ$(1,3)":INPUT #1,QZ$(1,1),QZ$(1,2),QZ$(1,3):PRINT:PRINT "SAVING RECORD #1":PRINT:PRINT "ENTER #1,QZ$(1,1),QZ$(1,2),QZ$(1,3)"
960 GOTO 950
970 CLS:PRINT "QUIZ FILE CONTENTS":PRINT:FOR Z=1 TO NR:PRINT Z;QZ$(Z,1):NEXT Z:PRINT:INPUT "ENTER RECORD # TO BE INSERTED";Z:PRINT:PRINT "ENTER #1,QZ$(Z,1),QZ$(Z,2),QZ$(Z,3)"
980 INPUT #1,QZ$(Z,1),QZ$(Z,2),QZ$(Z,3):PRINT:PRINT "SAVING RECORD #";Z$:PRINT:PRINT "ENTER #1,QZ$(Z,1),QZ$(Z,2),QZ$(Z,3)"
990 GOTO 980
1000 CLS:PRINT "QUIZ FILE CONTENTS":PRINT:FOR Z=1 TO NR:PRINT Z;QZ$(Z,1):NEXT Z:PRINT:INPUT "ENTER RECORD # TO BE DELETED";Z:PRINT:PRINT "ENTER #1,QZ$(Z,1),QZ$(Z,2),QZ$(Z,3)"
1010 INPUT #1,QZ$(Z,1),QZ$(Z,2),QZ$(Z,3):PRINT:PRINT "SAVING RECORD #";Z$:PRINT:PRINT "ENTER #1,QZ$(Z,1),QZ$(Z,2),QZ$(Z,3)"
1020 GOTO 1010
1030 CLS:GOTO 290
1040 '
1050 '
1060 '
1070 '
1080 '
1090 '
1100 A$=INKEY$:IF A$="" THEN 1100 ELSE RETURN
1110 '
1120 '
1130 '
1140 '
1150 '
1160 '
1170 '
1180 '
1190 '
1200 '
1210 '
1220 '
1230 '
1240 '
1250 '
1260 '
1270 '
1280 '
1290 '
1300 '
1310 '
1320 '
1330 '
1340 '
1350 '
1360 '
1370 '
1380 '
1390 '
1400 '
1410 '
1420 '
```

HCM

QUIZ-MAKE

TI-99/4A

```
100 REM ********QUIZ-MAKER*********
110 REM *********BY HOME WORLD*********
120 REM ********************************
130 REM HOME WORLD BASIC COMPUTER
140 REM VERSION 4.5.1
150 REM EXTENDED BASIC
160 REM
170 CALL CLEAR
180 CALL SCREEN(16)
190 CALL CHAR(40,"FFFFFFFFFFFFFFFF")
200 CALL CHAR(45,"0000000000000000")
210 CALL CHAR(42,"0000000F0F0F0F0F")
220 PRINT "QUIZ-MAKER":::"  BY":"HOME WORLD"
230 PRINT :::"ENTER":"1 TO MAKE A QUIZ":"2 TO EDIT A QUIZ":"3 TO PRINT A QUIZ":"4 TO SAVE A QUIZ":"5 TO LOAD A QUIZ":"6 TO EXIT"
240 GOSUB 3140
250 CALL CLEAR
260 PRINT :::"QUIZ-MAKER":"1) MAKE A QUIZ":"2) EDIT A QUIZ":"3) PRINT A QUIZ":"4) SAVE A QUIZ":"5) LOAD A QUIZ":"6) EXIT"
270 PRINT :::"ENTER YOUR CHOICE"
280 GOSUB 3140
290 IF K<49 OR K>54 THEN 280
300 CALL CLEAR
310 ON K-48 GOTO 320,1390,1540,1730,2090
320 '
330 '
340 '
350 '
360 '
370 '
380 IF K<>48 THEN 410
390 GOSUB 2120
400 '
410 NR=NR+1
420 CALL CLEAR
430 CALL CHAR(96,"1C3E7FFFFF7F3E1C")
440 CALL CHAR(97,"0000FFFFFFFF0000")
450 CALL CHAR(120,"0000FFFFFFFF0000")
460 CALL CHAR(121,"0000FFFFFFFF0000")
470 PRINT "EDIT MODE"
480 '
490 '
500 GOSUB 3050
510 '
520 '
530 '
540 GOSUB 3050
550 '
560 '
570 '
580 GOSUB 3050
590 '
600 '
610 GOSUB 3040
620 '
630 '
640 GOSUB 3040
650 '
660 GOSUB 3180
670 IF K=49 THEN 880
680 '
690 GOSUB 3230
700 IF K=49 THEN 880
710 IF K=50 THEN 250
720 '
730 '
740 '
750 '
760 '
770 '
780 '
790 '
800 '
810 GOSUB 2500
820 GOSUB 3280
830 '
840 '
850 '
860 GOSUB 3050
870 GOTO 650
```

Continued

```
 880 B=32880
 890 SUBB=K"SK=E
 900 ARCH FOR"
 910 IF K4UBB=30400
 920 XROOST=SUO10@
 930 GOOSTRSUU$ZKS=(QOS1@(QZ$(ZZ,1),COM$,1)
 940 GOOSTP=PTOOXSE@G$(QZ$(ZZ,2),COM$,1)
 950 IF$=7ULSBEG$(QZ$(ZZ,1)),1,28)<29 THEN 1130
 960 OR KOOSTSULSBEG$(QZ$(ZZ,1)),29,28)
 970 OR=SU1QULST$(ZZ,2)
 980 OR=SULSTR$(ZZ)
 990 OR=SU1ZBZ$ HR$=CHAR(3,19,32,4)
1000 OR=SU1ZBZ$ HR$=CHAR(3,19,32,4)
1010 OR=SU1ZBZ$ HR$=CHAR(3,19,32,4)
1020 OR=SU1ZBZ$ HR$=CHAR(3,19,32,4)
1030 OR=SU1ZBZ$ HR$=CHAR(3,19,32,4)
1040 OR=SU1ZBZ$ HR$=CHAR(3,19,32,4)
1050 OR=SU1ZBZ$ HR$=CHAR(3,19,32,4)
1060 OR=SU1ZBZ$ HR$=CHAR(3,19,32,4)
1070 OR=SU1ZBZ$ HR$=CHAR(3,19,32,4)
1080 OR=SU1ZBZ$ HR$=CHAR(3,19,32,4)
1090 OR=SU1ZBZ$ HR$=CHAR(3,19,32,4)
1100 OR=SU1ZBZ$ HR$=CHAR(3,19,32,4)
1110 OR=SU1ZBZ$ HR$=CHAR(3,19,32,4)
1120 OR=SU1ZBZ$ HR$=CHAR(3,19,32,4)
1130 OR=SU1ZBZ$ HR$=CHAR(3,19,32,4)
1140 OR=SU1ZBZ$ HR$=CHAR(3,19,32,4)
1150 OR=SU1ZBZ$ HR$=CHAR(3,19,32,4)
1160 OR=SU1ZBZ$ HR$=CHAR(3,19,32,4)
1170 OR=SU1ZBZ$ HR$=CHAR(3,19,32,4)
1180 OR=SU1ZBZ$ HR$=CHAR(3,19,32,4)
1190 OR=SU1ZBZ$ HR$=CHAR(3,19,32,4)
1200 OR=SU1ZBZ$ HR$=CHAR(3,19,32,4)
1210 OR=SU1ZBZ$ HR$=CHAR(3,19,32,4)
1220 OR=SU1ZBZ$ HR$=CHAR(3,19,32,4)
1230 OR=SU1ZBZ$ HR$=CHAR(3,19,32,4)
1240 OR=SU1ZBZ$ HR$=CHAR(3,19,32,4)
1250 OR=SU1ZBZ$ HR$=CHAR(3,19,32,4)
1260 OR=SU1ZBZ$ HR$=CHAR(3,19,32,4)
1270 OR=SU1ZBZ$ HR$=CHAR(3,19,32,4)
1280 OR=SU1ZBZ$ HR$=CHAR(3,19,32,4)
1290 OR=SU1ZBZ$ HR$=CHAR(3,19,32,4)
1300 OR=SU1ZBZ$ HR$=CHAR(3,19,32,4)
1310 OR=SU1ZBZ$ HR$=CHAR(3,19,32,4)
1320 OR=SU1ZBZ$ HR$=CHAR(3,19,32,4)
1330 OR=SU1ZBZ$ HR$=CHAR(3,19,32,4)
1340 OR=SU1ZBZ$ HR$=CHAR(3,19,32,4)
1350 OR=SU1ZBZ$ HR$=CHAR(3,19,32,4)
1360 OR=SU1ZBZ$ HR$=CHAR(3,19,32,4)
1370 OR=SU1ZBZ$ HR$=CHAR(3,19,32,4)
1380 OR=SU1ZBZ$ HR$=CHAR(3,19,32,4)
1390 OR=SU1ZBZ$ HR$=CHAR(3,19,32,4)
1400 OR=SU1ZBZ$ HR$=CHAR(3,19,32,4)
1410 OR=SU1ZBZ$ HR$=CHAR(3,19,32,4)
1420 OR=SU1ZBZ$ HR$=CHAR(3,19,32,4)
1430 OR=SU1ZBZ$ HR$=CHAR(3,19,32,4)
1440 OR=SU1ZBZ$ HR$=CHAR(3,19,32,4)
1450 OR=SU1ZBZ$ HR$=CHAR(3,19,32,4)
1460 OR=SU1ZBZ$ HR$=CHAR(3,19,32,4)
1470 OR=SU1ZBZ$ HR$=CHAR(3,19,32,4)
1480 OR=SU1ZBZ$ HR$=CHAR(3,19,32,4)
1490 OR=SU1ZBZ$ HR$=CHAR(3,19,32,4)
1500 OR=SU1ZBZ$ HR$=CHAR(3,19,32,4)
1510 OR=SU1ZBZ$ HR$=CHAR(3,19,32,4)
1520 OR=SU1ZBZ$ HR$=CHAR(3,19,32,4)
1530 OR=SU1ZBZ$ HR$=CHAR(3,19,32,4)
1540 OR=SU1ZBZ$ HR$=CHAR(3,19,32,4)
1550 OR=SU1ZBZ$ HR$=CHAR(3,19,32,4)
1560 OR=SU1ZBZ$ HR$=CHAR(3,19,32,4)
1570 OR=SU1ZBZ$ HR$=CHAR(3,19,32,4)
1580 OR=SU1ZBZ$ HR$=CHAR(3,19,32,4)
1590 OR=SU1ZBZ$ HR$=CHAR(3,19,32,4)
1600 OR=SU1ZBZ$ HR$=CHAR(3,19,32,4)
1610 OR=SU1ZBZ$ HR$=CHAR(3,19,32,4)
1620 OR=SU1ZBZ$ HR$=CHAR(3,19,32,4)
1630 OR=SU1ZBZ$ HR$=CHAR(3,19,32,4)
1640 OR=SU1ZBZ$ HR$=CHAR(3,19,32,4)
1650 OR=SU1ZBZ$ HR$=CHAR(3,19,32,4)
1660 OR=SU1ZBZ$ HR$=CHAR(3,19,32,4)
1670 OR=SU1ZBZ$ HR$=CHAR(3,19,32,4)
1680 OR=SU1ZBZ$ HR$=CHAR(3,19,32,4)
1690 OR=SU1ZBZ$ HR$=CHAR(3,19,32,4)
1700 OR=SU1ZBZ$ HR$=CHAR(3,19,32,4)
1710 OR=SU1ZBZ$ HR$=CHAR(3,19,32,4)
1720 OR=SU1ZBZ$ HR$=CHAR(3,19,32,4)
1730 OR=SU1ZBZ$ HR$=CHAR(3,19,32,4)
1740 OR=SU1ZBZ$ HR$=CHAR(3,19,32,4)
1750 OR=SU1ZBZ$ HR$=CHAR(3,19,32,4)
1760 OR=SU1ZBZ$ HR$=CHAR(3,19,32,4)
1770 OR=SU1ZBZ$ HR$=CHAR(3,19,32,4)
1780 OR=SU1ZBZ$ HR$=CHAR(3,19,32,4)
1790 OR=SU1ZBZ$ HR$=CHAR(3,19,32,4)
1800 OR=SU1ZBZ$ HR$=CHAR(3,19,32,4)
1810 OR=SU1ZBZ$ HR$=CHAR(3,19,32,4)
1820 OR=SU1ZBZ$ HR$=CHAR(3,19,32,4)
1830 OR=SU1ZBZ$ HR$=CHAR(3,19,32,4)
1840 OR=SU1ZBZ$ HR$=CHAR(3,19,32,4)
```

```
1850 "PRESS ENTER TO CONTINUE"
1860 RS=1
1870 "RECORD #;";R:QZ$(R,1)):QZ$(R,
1880
1890 INT:1/B
1900 >INT(R/4)THEN 1930
1910 <3140
1920 ERR=2 THEN 1880
1930 IF SS="D" THEN 2030
1940 DISPLAY AT(12,1):"SEQUENTIAL"
1950 "RANDOM";LEVEL$;" LETTER CLUES";TLIM;"
1960 INPUT "RECORD #";Z:QZ$(Z,1):QZ$(
1970
1980 IF Z<1 OR Z>20 THEN 1990
1990 CALL KEY(0,K,S):IF S=0 THEN 1990
2000 IF K=13 THEN 3000
2010 IF K=11 THEN 2860
2020
2030 "QUIZ TITLE:"::
2040 "AUTHOR'S NAME:"::
2050
2060 "QUESTIONS HEADER:"::
2070
2080 "ANSWERS HEADER:"::
2090
2100 "QUIZ TYPE:"::
2110 DOM=40+(K>50)THEN 2310
2120 CHAR(23,3,K)
2130 "LETTER CLUE PERCENTAGE(0-80
2140
2150 IF S=0 THEN 2350
2160 DISPLAY(0-99)::FOR
2170 RESTORE 2470
2180 RANDOMIZE
2190 "LETTER CLUES";TLIM;"SE
2200 GOSUB 2470
2210 DISPLAY AT(12,1):"THIS CORRECT (Y/N)"
2220 CALL KEY(0,K,S):IF S=0 THEN 2220
2230 IF K=89 THEN 2710
2240 IF K<>78 THEN 2510
2250 CALL CLEAR
2260 CALL CHAR(128,"0000000000000000")
2270 CALL HCHAR(1,1,128,768)
2280 CALL VCHAR(1,1,128,768)
2290 CALL KEY(0,K,S):IF S=0 THEN 2290
2300 IF K<>89 AND K<>78 THEN 2290
2310 CALL CLEAR
2320 CALL CHAR(128,"0000000000000000")
2330 CALL HCHAR(1,1,128,768)
2340 CALL VCHAR(1,1,128,768)
2350 CALL KEY(0,K,S):IF S=0 THEN 2350
2360 IF K<>89 AND K<>78 THEN 2350
2370 IF K=89 THEN 2710
2380 CALL CLEAR
2390 CALL CHAR(128,"0000000000000000")
2400 CALL HCHAR(1,1,128,768)
2410 CALL VCHAR(1,1,128,768)
2420 CALL KEY(0,K,S):IF S=0 THEN 2420
2430 IF K<>89 AND K<>78 THEN 2420
2440 IF K=89 THEN 2710
2450 CALL CLEAR
2460 CALL CHAR(128,"0000000000000000")
2470 CALL HCHAR(1,1,128,768)
2480 CALL VCHAR(1,1,128,768)
2490 CALL KEY(0,K,S):IF S=0 THEN 2490
2500 IF K<>89 AND K<>78 THEN 2490
2510 CALL CLEAR
2520 CALL CHAR(128,"0000000000000000")
2530 CALL HCHAR(1,1,128,768)
2540 CALL VCHAR(1,1,128,768)
2550 CALL KEY(0,K,S):IF S=0 THEN 2550
2560 IF K<>89 AND K<>78 THEN 2550
2570 IF K=89 THEN 2710
2580 CALL CLEAR
2590 CALL CHAR(128,"0000000000000000")
2600 CALL HCHAR(1,1,128,768)
2610 CALL VCHAR(1,1,128,768)
2620 CALL KEY(0,K,S):IF S=0 THEN 2620
2630 IF K<>89 AND K<>78 THEN 2620
2640 IF K=89 THEN 2710
2650 CALL CLEAR
2660 CALL CHAR(128,"0000000000000000")
2670 CALL HCHAR(1,1,128,768)
2680 CALL VCHAR(1,1,128,768)
2690 CALL KEY(0,K,S):IF S=0 THEN 2690
2700 IF K<>89 AND K<>78 THEN 2690
2710 CALL CLEAR
2720 CALL CHAR(128,"0000000000000000")
2730 CALL HCHAR(1,1,128,768)
2740 CALL VCHAR(1,1,128,768)
2750 CALL KEY(0,K,S):IF S=0 THEN 2750
2760 IF K<>89 AND K<>78 THEN 2750
2770 IF K=89 THEN 2710
2780 CALL CLEAR
2790 CALL CHAR(128,"0000000000000000")
2800 CALL HCHAR(1,1,128,768)
2810 CALL VCHAR(1,1,128,768)
2820 CALL KEY(0,K,S):IF S=0 THEN 2820
2830 IF K<>89 AND K<>78 THEN 2820
2840
```

Continued

PROGRAM LISTING

QUIZ MAKE *Continued*

TI-99/4A

```
2850  GOTO 2980
2860  CALL KEY(XR,YR,K)
2870  IF K<>0 THEN 2970
2880  IF K=0 THEN 2750
2890  YR=YR-1
2900  GC=GC-1
2910  A$=SEG$(A$,1,LEN(A$)-1)
2920  IF YR>2 THEN 2950
2930  YR=30
2940  XR=XR-1
2950  CALL HCHAR(XR,YR,32)
2960  GOTO 2750
2970  YR=YR+1
2980  A$=A$&CHR$(K)
2990  NEXT Z
3000  CALL KEY(0,K,S)
3010  IF S<>1 THEN 3000
3020  RETURN
3030  REM  YOU NEED TO LOAD OR CREATE OP
      TION FILE BEFORE USING THIS OP
      TION!
3040  CALL SOUND(300,110,0)
3050  FOR T=1 TO 800
3060  NEXT T
3070  RETURN
3080  REM  YOU NEED TO LOAD OR CREATE OP
      TION FILE BEFORE USING THIS OP
      TION!
3090  CALL SOUND(300,110,0)
3100  FOR T=1 TO 800
3110  NEXT T
3120  GOTO 2750
3130  CALL KEY(0,K,S)
3140  IF S=0 THEN 3140
3150  IF K<49 OR K>53 THEN 3140
3160  RUN "CS1"
3170
3180
3190
3200  CALL HCHAR(7,1,32,64)
3210  CALL HCHAR(8,1,32,64)
3220
3230
3240
3250  CALL HCHAR(13,1,32,32)
3260
3270
3280  CALL HCHAR(4,1,32,12)
3290  CALL HCHAR(7,1,32,64)
3300  CALL HCHAR(10,1,32,12)
3310  CALL HCHAR(13,1,32,32)
3320  CALL HCHAR(18,1,32,32)
3330  CALL HCOUND(40,440,10)
3340
3350  CALL LEAR
3360  IF K<>89 AND K<>78 THEN 3360
3370  PRINT "YOU HAVE CHANGED THE QUIZ
      IT. WISH TO EXIT AND
      SAVE THOSE CHANGES (Y/N)"
3380  IF K=89 THEN 3430
3390  GOSUB 3400
3400  OPEN #1:"DSK1.QUIZ",OUTPUT
3410  PRINT #1:A$
3420  CLOSE #1
3430  STOP
3440  END
```

HCM

QUIZ TAKE

APPLE // Family

```
100   REM  ********************
110   REM  *  QUIZMAKER.BASIC  *
120   REM  *  BY ANDY HOPPE    *
130   REM  *  COPYRIGHT 1983   *
140   REM  *  BY COMPUTE!      *
150   REM  *  VERSION 4.5.1    *
160   REM  *  FOR APPLESOFT    *
170   REM  ********************
180
190   HOME : GOTO 1330
200   DIM QZ$(500,2),WC$(5),ER$(15)
210   FOR I = 1 TO 15: READ ER$(I): NEXT
220   VTAB 24: HTAB 13: PRINT "QUIZ MA
      KER": PRINT : PRINT : PRINT "BY
      ANDY HOPPE": PRINT : PRINT "COPY
      RIGHT 1983": PRINT "BY COMPUTE!
      ": PRINT : PRINT "VERSION 4.5.1
      ": PRINT : PRINT "FOR APPLESOFT
      ": FOR I = 1 TO 2000: NEXT I
230   HOME : VTAB 8: PRINT "DO YOU WIS
      H TO": PRINT "(C)REATE A NEW QU
      IZ": PRINT "(L)OAD AN OLD QUIZ"
      : PRINT "(E)DIT AN OLD QUIZ": P
      RINT "(T)AKE A QUIZ": PRINT "(Q
      )UIT": PRINT : PRINT "PLEASE TY
      PE C, L, E, T, OR Q"
240   GET A$: IF A$ = "C" THEN 1330
      : IF A$ = "L" THEN 1330: IF A$
      = "E" THEN 1330: IF A$ = "T" TH
      EN 1330
250   IF A$ = "Q" THEN 2600
260   GOTO 240
270   REM  ********************
      *  TAKE.QUIZ.BASIC  *
      ********************
280   REM  *  BY ANDY HOPPE    *
290   REM  *  COPYRIGHT 1983   *
300   REM  *  BY COMPUTE!      *
310   REM  *  VERSION 4.5.1    *
320   REM  *  FOR APPLESOFT    *
330   REM  ********************
340   GOSUB 1330
350   FOR I = 1 TO 5: PRINT : NEXT I:
      VTAB 24: HTAB 13: PRINT "TAKE
      QUIZ": PRINT : PRINT : PRINT "
      BY ANDY HOPPE": PRINT : PRINT "
      COPYRIGHT 1983": PRINT "BY COMP
      UTE!": PRINT : PRINT "VERSION 4
      .5.1": PRINT : PRINT "FOR APPLE
      SOFT": FOR I = 1 TO 2000: NEXT
      I
360   HOME : VTAB 8: PRINT "DO YOU WI
      SH TO": PRINT "(C)REATE A NEW Q
      UIZ": PRINT "(L)OAD AN OLD QUIZ
      ": PRINT "(E)DIT AN OLD QUIZ":
      PRINT "(T)AKE A QUIZ": PRINT "(
      Q)UIT": PRINT : PRINT "PLEASE T
      YPE C, L, E, T, OR Q"
370   GET A$: IF A$ = "C" THEN 1330
      : IF A$ = "L" THEN 1330: IF A$
      = "E" THEN 1330: IF A$ = "T" TH
      EN 340
380   IF A$ = "Q" THEN 2600
390   GOTO 370
400   REM  ********************
      *  TAKE.QUIZ.BASIC  *
      ********************
410   REM  *  BY ANDY HOPPE    *
420   REM  *  COPYRIGHT 1983   *
430   REM  *  BY COMPUTE!      *
440   REM  *  VERSION 4.5.1    *
450   REM  *  FOR APPLESOFT    *
460   REM  ********************
470   REM
480   REM
490   REM  BALTHROP
      MAGAZINE
500   REM
510   REM
520   REM  READ ERROR MESSAGES
530   FOR I = 1 TO 15: READ ER$(I): NEXT
      I
540   REM  SET UP VARIABLES
550   REM  ********************
560   REM  *  QUIZ VARIABLES  *
570   REM  ********************
580   REM  *  QZ$(500,2)      *
590   REM  *  WC$(5)          *
600   REM  *  ER$(15)         *
610   REM  ********************
620   REM
630   REM
640   REM
650   REM
660   REM
670   REM
680   REM
690   REM
700   REM
710   REM
720   REM
730   REM
740   REM
750   REM
760   REM
770   REM
780   REM
790   REM
800   REM
810   REM
```

Continued

QUIZ-TAKE

APPLE // Family

```
 820 
 830 
 840 
 850 
 860 
 870 
 880 
 890 
 900 
 910 
 920 
 930 
 940 
 950 
 960 
 970 
 980 
 990 
1000 
1010 
1020 
1030 
1040 
1050 
1060 
1070 
1080 
1090 
1100 
1110 
1120 
1130 
1140 
1150 
1160 
```

```
1170 
1180 
1190 
1200 
1210 
1220 
1230 
1240 
1250 
1260 
1270 
1280 
1290 
1300 
1310 
1320 
1330 
1340 
1350 
1360 
1370 
1380 
1390 
1400 
1410 
```

HCM

QUIZ-TAKE *Continued*

COMMODORE 64

```
 100 
 110 
 120 
 130 
 140 
 150 
 160 
 170 
 180 
 190 
 200 
 210 
 220 
 230 
 240 
 250 
 260 
 270 
 280 
 290 
 300 
 310 
 320 
 330 
 340 
 350 
 360 
```

```
 370 
 380 
 390 
 400 
 410 
 420 
 430 
 440 
 450 
 460 
 470 
 480 
 490 
 500 
 510 
 520 
 530 
 540 
 550 
 560 
 570 
 580 
 590 
 600 
```

Continued

```
 610 PRINT:PRINTQT$
 620 FOR I=1 TO 40:PRINT"■SHIFT *■";:NEX
 630 PRINT:PRINT
 640 FOR I=1 TO 40:PRINT"■SHIFT *■";:NEX
 650 PRINT:PRINT:PRINT:PRINTAT$
 660 FOR I=1 TO 40:PRINT"■SHIFT *■";:NEX
 670 PRINT
 680 FOR I=1 TO 40:PRINT"■SHIFT *■";:NEX
 690 X=5:Y=24:GOSUB2190:PRINT"■CTRL RVSO
     NL■ RVSOFF■";
 700 RETURN
 710 DISPLAY QUESTION AND GET ANSWER
 720 RG=0:RG=TG:WG=TG:SG=TG:LV=VAL(K$)
 730 PR=0:PR=PR+1:IF PR>NR THEN RETURN
 740 IF QT=2 THEN PR=INT(RND(0)*NR)+1:GO
     TO 760
 750 PR=PR+1:IF PR>NR THEN RETURN
 760 WA=0:GOSUB 1900:X=0:Y=8:GOSUB2190:PR
     INT(QZ$(PR,1));:GOSUB2010
 770 IF LV=3 THEN GOSUB2210
 780 X=0:Y=16:GOSUB2190:LL=39:KL=32:KU=9
     0:GOSUB2210
 790 IF ASC(K$)=216 THEN PR=NR+1:RETURN
 800 REM ENTER GOTO AP PROVE AND BRANCH
     DEP$=ST$:IF ANS$=QZ$(PR,2) THEN 1260
 810 IF LV=1 AND WA=0 THEN 1210
 820 IF LV=3 AND WA=1 THEN 1080
 830 IF LV=3 AND WA<2 THEN 970
 840 IF LV=3 AND WA<2 THEN 1080
 850 IF LV=4 AND WA<1 THEN 970
 860 IF LV=4 AND WA=1 THEN 1080
 870 IF LV=2 OR LV=3 THEN 1080
 880 REM WRONG ANSWER ROUTINE
 890 GOSUB2150:I=20:POKE 54296,15:FOR J=1
     TO 20:POKE 54296,0
 900 FOR I=1 TO 20:POKE 54296,0:NEXT:NEXT
     :GOTO 740
 910 IF I=1 TO 20:POKE 54296,15:FOR J=1
     TO 20:POKE 54296,0
 920 FOR I=1 TO 20:POKE 54296,0:NEXT:NEXT
 930 X=6:Y=19:GOSUB2190:PRINT"<WRONG> TH
     E RIGHT ANSWER IS"
 940 X=(40-LEN(QZ$(PR,2)))/2:Y=21:GOSUB2
     190:PRINTQZ$(PR,2)
 950 FOR I=1 TO 1000:NEXT:GOSUB2120
 960 WG=WG+1:TG=TG+1:GOTO740
 970 REM DISPLAY CLUES
 980 FOR I=1 TO 20:POKE 54296,0:NEXT:NEXT
     :POKE 54296,15:FOR J=1
 990 FOR E=1 TO 3:NEXT:NEXT
1000 POKE 781,16:SYS 59903
1010 PL=LEN(QZ$(PR,2)):NL%=L*LCP/100:ST$=
     "■"
1020 FOR I=1 TO L:ST$=ST$+"*":NEXT
1030 FOR I=1 TO NL%
1040 R%=RND(0)*L+1:U$=MID$(QZ$(PR,2),R%,
     1):ST$=LEFT$(ST$,R%-1)+U$+MID$(ST$,R%+
     1):NEXT
1050 X=0:Y=16:GOSUB2190:PRINTST$:WA=WA+1
     :GOTO780
1060 REM CHECK SPELLING
1070 PP=0:PP$=QZ$(PR,2):A$=P$:B$=AN$
1080 IF LEN(B$)>LEN(P$) THEN B$=P$:A$=AN
     $
1090 FOR I=1 TO LEN(A$)
1100 IF MID$(A$,I,1)<>MID$(B$,I,1) THEN
     PP=PP+1
1110 NEXT
1120 IF (LEN(P$)-PP)/LEN(P$)<.7 THEN 890
1130 GOSUB2150:X=19:GOSUB2190:PRINT"MISSP
     ELLED WORD"
1140 X=5:Y=19:GOSUB2190:PRINT"■SPELLING
     IS■ RIGHT"
1150 SG=SG+1:XT=(40-LEN(P$))/2:Y=21:GOSUB
     2190:PRINTP$
1160 FOR I=1 TO 1000:NEXT
1170 GOSUB2120:GOTO780
1180 REM WORD SCRAMBLE
1190 FOR I=1 TO LEN(A$)
1200 MID$(A$,I,1)="■":NEXT
1210 GOSUB2155:GOTO780
1220 REM TRY AGAIN
1230 POKE 781,16:SYS 59903
1240 FOR I=1 TO 20:POKE 54296,15:FOR J=1
     TO 20:POKE 54296,0
1250 FOR E=1 TO 3:NEXT:NEXT
1260 WA=WA+1:GOTO780
1270 REM RIGHT ANSWER
1280 GOSUB2150
1290 FOR I=B TO B+24:POKE I,0:NEXT:POKE
     B+5,85:POKE B+6,85:POKE B+12,8
1300 POKE B+24,15:POKE B+4,33:POKE B+11,
     17
1310 FOR J=1 TO 6:READ H1,L1,H2,L2:POKE
     B+1,H1:POKE B,L1:POKE B+8,H2:POKE B+7,
     L2
1320 FOR H=1 TO 50:FOR E=1 TO 200:NEXT
1330 NEXT:GOSUB2120:GOTO780
1340 REM DATA
1350 DATA 25,30,18,209,33,135,25,30,42,6
     2,31,165,50,60,37,162,42,62,31,165,50
     ,60
1360 DATA 60,37,162
1370 NEXT:POKE B+4,32:POKE B+11,16:FOR E
     =1 TO 500:NEXT
1380 REM RESTORE
1390 FOR I=B TO B+24:POKE I,0:NEXT
1400 X=9:Y=19:GOSUB2190:PRINT"■*■*■ THAT I
     S■ RIGHT ■*■*■"
1410 FOR I=1 TO 1000:NEXT
1420 RG=RG+1:GOSUB2120:GOTO740
1430 REM INTRO QUIZ FILE
1440 PRINT"■SHIFT CLR■":VIN=1:TG=0
1450 PRINT"■ LOAD DATA":PRINT:PRINT
     :PRINT"■ NAME :";:LL=14:KL=65:KU=
     90:GOSUB2210:PRINT
1460 X$=STR$(TAB(PRIT):PRINT:FIB(14)
     ):GOSUB2190:PRINT"■CTRL RVSOFF■USE
     RNAME";:GOSUB2155:GOSUB1550:RETURN
1470 X$=STR$(TAPE OR DISK (T/D) ";:
1480 PGOSUB2340:IF K$<>"T" AND K$<>"D" TH
     EN1490
1490 DV$="■CTRL RVSON■TAPE":IF K$="D" TH
     EN DV$="■CTRL RVSON■DISK"
1500 PRINT:DV$=":PRINT"■CTRL RVSON■TAPE"
     :IF DV$=".Q" THEN1540
1510 PRINT:OPEN15,8,15,"I":CLOSE15:OPEN1
     ,8,0,"$":PRINT"■CTRL RVSON■TAPE"
1520 OPEN1,8,0,"■CTRL RVSON■$":PRINT"■CT
     RL RVSON■DISK"
1530 OPEN15,8,15,"I":OPEN1,8,0,"$":PRINT
     "■CTRL RVSON■TAPE"
1540 OPEN1,1,0,"Q":GOSUB1550:RETURN
1550 INPUT#1,TL$
1560 INPUT#1,DT$
1570 INPUT#1,CNT$
1580 INPUT#1,AT$
1590 INPUT#1,NR
1600 INPUT#1,CT
1610 INPUT#1,LD:IF TL$="" THEN VIN=0:GOT
     O1700
1620 PRINT"LAST MODIFIED ON "DT$:PRINT"N
     UMBER OF QUESTIONS :"CNT$:PRINT"QUES
     TION HEADER :"AT$:PRINT"ANSWER HEADER
     :"NR" RECORDS":IF NR=0 THEN I=1:GOTO
     1700
1630 FOR I=1 TO NR
1640 INPUT#1,QZ$(I,1)
1650 INPUT#1,QZ$(I,2):NEXT
1660 CLOSE15:PRINT"■CTRL RVSON■TAPE THEN
     1720
1670 FOR I=1 TO NR
1680 INPUT#1,QZ$(I,1)
1690 INPUT#1,QZ$(I,2):NEXT
1700 DV$="■CTRL RVSON■TAPE":IF DV$=".Q" T
     HEN1720
1710 OPEN15,8,15:INPUT#15,V,S$:CLOSE15:I
     F V<>0 THEN PRINT:PRINTS$:GOTO1720
1720 X=5:Y=24:GOSUB2190:PRINT"■CTRL RVSO
     NL■ PRESS■ ANY■ KEY TO CONTINUE";
1730 GOSUB2210:IF ASC(K$)<>13 THEN1730
1740 CLOSE1:RETURN
1750 REM INSTRUCTIONS
1760 PRINT"■SHIFT CLR■■CTRL RVSOFF■USE
     UP QUIZ■ DOWN ARROW KEYS TO SCROLL
     ANSWERS SHIFT X WHEN YOU ARE DONE":PR
     INT"PRESS RETURN TO START":
1770 QUIN:DONE":PRINT"PRESS RETURN TO STA
     RTE";:ZZ$:IF ZZ$="" THEN1780
1780 PRINT:RETURN
1790 REM SHIFT CLR■ :R=1:X=0:Y=24:GOSUB2
     190:PRINT"■SHIFT CLR■";
1800 FOR I=1 TO 98,0:FOR I=1 TO 40:PRINT"
     ■SHIFT C■";:NEXT:PRINT"■SHIFT C■";:
     NEXT
1810 PRINT:QZ$(R,1):PRINT QZ$(R,2)
1820 FOR I=1 TO 40:PRINT"■SHIFT C■";:NEX
     T:PRINT:GOSUB2310
1830 GOSUB2340
1840 IF ASC(K$)=17 AND R<NR THEN1880
1850 IF ASC(K$)=145 AND R>1 THEN1890
1860 IF ASC(K$)=216 THEN RETURN
1870 GOTO1830
1880 R=R+1:GOTO1800
1890 R=R-1:GOTO1800
1900 REM DISPLAY THE SCORES
1910 Y=4
1920 IF TG=0 THEN2000
1930 X=2:GOSUB2190:PRINT"■CTRL RVSOFF■R
     .WR.SP.";
1940 ST$=STR$(INT((RG/TG*100)))+"%"
1950 X=10-LEN(ST$):GOSUB2190:PRINTST$
1960 X=24-LEN(ST$):GOSUB2190:PRINTST$
1970 X=38-LEN(ST$):GOSUB2190:PRINTST$
1980 X=24-LEN(ST$):GOSUB2190:PRINTST$
1990 GOSUB2190:PRINTST$
2000 RETURN
2010 REM CHOOSE FIVE RANDOM WORDS FOR TH
     E QUIZ
2020 FOR I=1 TO 5
2030 WC%=RND(0)*(NR)+1:IF WC%=PR THEN 2
     030
2040 WC$=QZ$(WC%,2)
2050 NEXT
2060 WC$=STR$(INT(RND(0)*5)+1)=QZ$(PR,2)
2070 X=0:Y=19:GOSUB2190
2080 FOR I=1 TO 5
2090 PRINTWC$(I)
2100 NEXT
2110 RETURN
2120 REM CLEAR THE QUESTION AND ANSWER F
     IELDS
2130 POKE781,8:SYS 59903:POKE 781,9:SYS
     59903
2140 POKE781,16:SYS 59903
2150 POKE781,19:SYS 59903:POKE 781,20:S
     YS 59903:POKE781,21:SYS 59903
2160 POKE781,20:SYS 59903:POKE 781,22:SY
     S 59903
2170 POKE781,23:SYS 59903
2180 RETURN
2190 REM ROUTINE TO SIMULATE THE PRINT A
     T STATEMENT
```

Continued

QUIZ-TAKE *Continued*

COMMODORE 64

```
2200 POKE 781,Y:POKE 782,X:POKE 783,0:SYS 65520:RETURN
2210 REM INPUT ROUTINE
2220 POKE 198,0:L=0:ST$=""
2230 PRINT"{CMDR P}{SHIFT CRSRLEFT}";:GOSUB2340
2240 IF ASC(K$)=13 AND L<>0 THEN PRINT" ":GOTO2330
2250 IF ASC(K$)=216 AND LL=39 THEN2330
2260 IF ASC(K$)<>20 OR L=0 THEN2280
2270 ST$=LEFT$(ST$,LEN(ST$)-1):PRINT" {2 SHIFT CRSRLEFT}";:L=L-1:GOTO2230
2280 IF ASC(K$)<KL OR ASC(K$)>KU THEN2230
2290 IF ASC(K$)=34 THEN2230
2300 PRINTK$;:ST$=ST$+K$:L=L+1
2310 IF L=LL THEN2330
2320 GOTO2230
2330 RETURN
2340 REM SINGLE KEY INPUT
2350 GET K$:IF K$="" THEN2350
2360 POKE 198,0:RETURN
2370 REM ILLEGAL ENTRY MESSAGE
2380 PRINT"{SHIFT CLR}":X=5:Y=10:GOSUB2190
2390 PRINT"YOU NEED TO LOAD A QUIZ FILE":PRINT
2400 PRINTTAB(7)"INTO MEMORY BEFORE USING":PRINT
2410 PRINTTAB(13)"THIS OPTION"
2420 FOR I=1 TO 3250:NEXT:RETURN
2430 REM EXIT PROGRAM ROUTINE
2440 PRINT"{SHIFT CLR}":PRINT"BYE BYE, SEE YOU NEXT TIME..."
2450 POKE 53280,14:POKE 53281,6:POKE 646,14:POKE 650,0:POKE 657,0:END
```

HCM

QUIZ-TAKE

IBM PC & IBM PCjr

```
100 '  *************
110 '  *  QUIZ-TAKE  *
120 '  *************
130 '  BY WILLIAM K. BALTHROP
140 '  HOME COMPUTER MAGAZINE
150 '  VERSION 4.5.1
160 '  IBM PCjr WITH CARTRIDGE BASIC
170 '  FROM DOS 2.1
180 '  IBM PC WITH BASICA
190 '  INITIALIZE PROGRAM--TITLE SCREEN
200 '
210 ON ERROR GOTO 940
220 SCREEN 0:DEFINT A-Z:RANDOMIZE TIMER:KEY OFF:MXREC=600:DIM QZ$(MXREC,2),ERM$(9),ERCD(9):VIN=0:KEY 1,CHR$(10):KEY 2,CHR$(11)
230 BL$=STRING$(40,32):RESTORE 1060:FOR Z=1 TO 9:READ ERCD(Z),ERM$(Z):NEXT
240 CLS:LOCATE 12,13:PRINT "QUIZ-TAKE":LOCATE 20,9:PRINT "PRESS [Esc] TO EXIT":PRINT:PRINT TAB(9);"PRESS [";:PRINT CHR$(17);CHR$(196);CHR$(217);:PRINT "] TO START"
250 A$=INKEY$:IF A$=CHR$(27) THEN 1100 ELSE IF A$<>CHR$(13) THEN 250
260 '
270 '  MAIN MENU
280 '
290 CLS:LOCATE 1,13:PRINT "QUIZ-TAKE":LOCATE 5,10:PRINT "** MAIN MENU **":LOCATE 7,1:PRINT "1)     TAKE QUIZ":PRINT:PRINT "2)     LOAD QUIZ":PRINT:PRINT "3)     STUDY QUIZ":PRINT:PRINT "4)     EXIT":IF TOTG>0 THEN GOSUB 1140
300 GOSUB 800:IF A$<"1" OR A$>"4" THEN 300 ELSE ON VAL(A$) GOTO 340,700,620,240
310 '
320 '  TAKE THE QUIZ
330 '
340 CLS:IF VIN=0 THEN LOCATE 12,1:PRINT "YOU NEED TO LOAD A QUIZ FILE FROM DISK.":PRINT:PRINT "USE OPTION # 2 ON THE MAIN MENU.":PRINT:PRINT "PRESS [ENTER] TO CONTINUE":SOUND 220,10:GOSUB 800:GOTO 290
350 LOCATE 1,12:PRINT "** QUIZ-TAKE **":LOCATE 4,1:PRINT "1)     WORD CLUES / 2 TRIES":PRINT:PRINT "2)     WORD CLUES / 1 TRY":PRINT:PRINT "3)     LETTER CLUES / 3 TRIES":PRINT:PRINT "4)     LETTER CLUES / 2 TRIES":PRINT:PRINT "5)     NO CLUES / 1 TRY"
360 PRINT:PRINT "6)     SAME QUIZ":PRINT:PRINT "7)     EXIT":IF TOTG>0 THEN GOSUB 1140
370 GOSUB 800:IF A$<"1" OR A$>"7" THEN 370 ELSE IF A$="7" THEN GOTO 290 ELSE LE=VAL(A$)
380 IF LE<6 THEN TOTG=0:RG=0:WG=0:SG=0:PROB=0:LEV=LE
390 IF LEV=0 THEN CLS:PRINT "YOU MUST START A QUIZ WITH LEVELS 1 THROUGH 5 BEFORE USING THIS OPTION.":PRINT:PRINT "PRESS [ENTER] TO CONTINUE":SOUND 220,10:GOSUB 800:GOTO 340
400 CLS:IF QT=1 THEN PROB=PROB+1:IF PROB>NR THEN PROB=0:GOTO 400 ELSE ELSE PROB=RND*NR+1:IF PROB>NR THEN 400
410 LOCATE 2,20-(LEN(TTL$)/2):PRINT TTL$:IF TOTG>0 THEN PRINT USING "RIGHT:###%     WRONG:###%     SPELL:###%";RG/TOTG*100,WG/TOTG*100,SG/TOTG*100
420 LOCATE 1,12:PRINT "** QUIZ-TAKE **":LOCATE 4,1:PRINT QT$:PRINT STRING$(40,205);:LOCATE 8,1:PRINT STRING$(40,205);:LOCATE 10,1:PRINT AT$:PRINT STRING$(40,205);:LOCATE 13,1:PRINT STRING$(40,205);:LOCATE 25,1:PRINT "PRESS [ENTER] TO RETURN TO MENU";
430 IF LEV>2 THEN 480
440 FOR WC=1 TO 5
450 W=RND*NR+.99:IF W=PROB OR QZ$(W,2)="" OR W>NR OR W<1 THEN 450 ELSE WC$(WC)=QZ$(W,2)
460 NEXT
470 W=RND*5+.99:IF W<1 OR W>5 THEN 470 ELSE WC$(W)=QZ$(PROB,2):FOR Z=1 TO 5:LOCATE Z+16,1:PRINT WC$(Z);:NEXT
480 LOCATE 6,1:PRINT LEFT$(QZ$(PROB,1),40);:LOCATE 7,1:PRINT MID$(QZ$(PROB,1),41,40);:LOCATE 12,1:ML=40:GOSUB 840:IF A$="" THEN GOTO 290 ELSE ANS$=A$
490 IF ANS$=QZ$(PROB,2) THEN 580 ELSE IF (LEV=3 AND WA<2) OR (LEV=4 AND WA<1) THEN GOSUB 550:WA=WA+1:GOTO 480 ELSE IF LEV=1 AND WA<1 THEN LOCATE 12,1:PRINT BL$;:WA=WA+1:SOUND 110,5:GOTO 480
500 GOSUB 520:IF SPCK<.7 THEN LOCATE 14,1:PRINT "THAT IS INCORRECT--THE ANSWER IS:":PRINT QZ$(PROB,2);:SOUND 220,10:WG=WG+1:TOTG=TOTG+1:WA=0:FOR TD=1 TO TLIM*1000:NEXT:GOTO 400
510 LOCATE 14,1:PRINT "INCORRECT SPELLING--THE ANSWER IS:":PRINT QZ$(PROB,2);:SOUND 330,5:SG=SG+1:TOTG=TOTG+1:WA=0:FOR TD=1 TO TLIM*1000:NEXT:GOTO 400
520 IF LEN(ANS$)>LEN(QZ$(PROB,2)) THEN SP=LEN(ANS$) ELSE SP=LEN(QZ$(PROB,2))
530 SPC=0:FOR Z=1 TO SP:IF MID$(ANS$,Z,1)=MID$(QZ$(PROB,2),Z,1) THEN SPC=SPC+1
540 NEXT:SPCK=SPC/SP:RETURN
550 NL=LEN(QZ$(PROB,2))*LCP/100:IF NL<1 THEN RETURN ELSE LOCATE 12,1:PRINT STRING$(LEN(QZ$(PROB,2)),254);STRING$(40-LEN(QZ$(PROB,2)),32);
560 QL=LEN(QZ$(PROB,2)):FOR Z=1 TO NL
570 CP=RND*QL+1:IF CP>40 OR CP<1 THEN 570 ELSE LOCATE 12,CP:PRINT MID$(QZ$(PROB,2),CP,1);:NEXT:RETURN
580 LOCATE 14,1:PRINT "!!!!! THAT IS RIGHT !!!!!":PLAY "MBL32O2CDEFGABAGFGABO3CDEFGABAGFGABO4CDEFGABAGFGAB":GOSUB 1020:RG=RG+1:TOTG=TOTG+1:WA=0:GOTO 400
590 '
600 '  STUDY MODE
610 '
620 CLS:IF VIN=0 THEN LOCATE 12,1:PRINT "YOU NEED TO LOAD A QUIZ FILE IF":PRINT "YOU WANT TO USE THE STUDY MODE.":PRINT:PRINT "SELECT OPTION #2 FROM THE MAIN MENU":PRINT:PRINT "PRESS [ENTER] TO CONTINUE":SOUND 220,10:GOSUB 800:GOTO 290
630 LOCATE 25,1:PRINT "[Fn 1]--UP     [Fn 2]--DOWN     [Esc]--EXIT";:RN=1
640 LOCATE 1,1:FOR Z=1 TO 4:PRINT QZ$(Z,1):PRINT QZ$(Z,2):PRINT:PRINT:PRINT:NEXT
650 GOSUB 800:IF A$=CHR$(11) AND RN<NR THEN RN=RN+1:GOSUB 660:GOTO 650 ELSE IF A$=CHR$(10) AND RN>1 THEN RN=RN-1:GOSUB 660:GOTO 650 ELSE IF A$=CHR$(27) THEN 290 ELSE SOUND 220,1:GOTO 650
660 PRINT QZ$(RN,1):PRINT QZ$(RN,2):PRINT:PRINT:PRINT:RETURN
670 '
680 '  LOAD QUIZ FILE
690 '
700 CLS:LOCATE 1,10:PRINT "** LOAD QUIZ FILE **"
710 LOCATE 5,1:PRINT "QUIZ FILE NAME: [          ]":LOCATE 5,18:ML=10:GOSUB 840
720 A$=A$+".QZ":OPEN A$ FOR INPUT AS #1
```

Continued

QUIZ-TAKE *Continued*

IBM PC & IBM PCjr

```
730 INPUT #1,TTL$,DT$,CN$,QT
    ,LLIM:LOCATE 8,20-LEN(TT$),ATL$,NR,QT
    RIN":PRINT "HERE ARE ";NR;" RECORD
    S.":CLOCATE 20,1:PRINT "READING RECORD #
740 ";:LOCATE 20,1:PRINT "READING RECORD #
750 FOR Z=1 TO NR:INPUT #1,QZ$(Z,1),QZ$
    (Z,2):LOCATE 20,17:PRINT Z:NEXT
760 VINT 1:CLOSE #1:LOCATE 23,1:PRINT:PR
    INT"PRESS [ENTER] TO CONTINUE":GOSUB 290
770
780 ' SINGLE KEY - INPUT ROUTINE
790
800 A$=INKEY$:IF A$="" THEN 800 ELSE RE
    TURN
810
820 ' MULTIPLE KEY - INPUT ROUTINE
830
840 XP=POS(0):YP=CSRLIN:A$="":DEF SEG=0
    :POKE 1050,PEEK(1052)
850 LOCATE ,,1:YP,XP,1,0,7:K$=INKEY$:IF K$=
    "" THEN 850
860 IFK$=CHR$(13) THEN PRINT CHR$(32);
    :LOCATE ,,0:RETURN
870 IFK$=CHR$(8) THEN PRINT CHR$(32);:
    XP=XP-1:IF XP<1 THEN XP=40:YP=YP-1
880 IFINT(K$)=CHR$(8);:A$=LEFT$(A$,LEN(A$)-1)
    :GOTO 850
890 A$=A$+K$:PRINT K$;:IF LEN(A$)=ML THEN
    K$=MID$(A$,LEN(A$),1):GOTO 880
900 IF XP=40 THEN XP=1:YP=YP+1
910 GOTO 850
920 ' ERROR ROUTINES
930 '
940 CLOSE:LOCATE 25,1:R=ERR:L=ERL:FOR Z
    =1 TO 9:IF ERCD(Z)=R THEN 960
950 NEXT:PRINT "ERROR #";R;" IN LINE #"
    ;L;:GOTO 970
960 PRINT ERM$(Z);" #";R;
970 FOR TD=1 TO 10000:NEXT:RETURN
980 :LOCATE 25,1:PRINT STRING$(39,32);
    :RESUME 290
990
1000 ' TIME DELAY
1010
1020 FOR TD=1 TO TLIM*1000:NEXT:RETURN
1030
1040 ' PROGRAM DATA
1050
1060 DATA 64,BAD FILE NAME,25,DEVICE FAU
     LT,57,DEVICE I/O ERROR,24,DEVICE TIME
     OUT,68,DEVICE UNAVAILABLE,72,DISK ME
     DIA ERROR,71,DISK NOT READY,53,FILE
     NUMBER OR NAME
1070
1080 ' END OF PROGRAM ROUTINE
1090
1100 CLS:PRINT "BYE-BYE":PRINT:PRINT "S
     EE YOU NEXT TIME":END
1110
1120 ' DISPLAY SCORE
1130
1140 LOCATE 20,1:PRINT "RIGHT:";INT(R
     G/TOTG*100);"%":PRINT "WRONG:";INT(W
     G/TOTG*100);"%":PRINT "SPELLING:";INT(S
     G/TOTG*100);:RETURN
```

HCM

QUIZ-TAKE

TI-99/4A

```
100 REM ***[UIZ-TAKE]***
110 REM *** QUIZ-TAKE ***
120 REM ***  BY WILLIAM  ***
130 REM ***  BALTHROP    ***
140 REM ***  HOMESOFT    ***
150 REM ***  VERSION 5.1 ***
160 REM ***  TI BASIC    ***
170 REM ***  EXTENDED BASIC***
180 REM
190 CALL CLEAR
200 RANDOMIZE
210 OPTION BASE 1
220 DIM Q$(40,2),WC$(5)
230 CALL CHAR(45,"0000FFFFFFFF")
240 RIN$="  ":: ARTB$="  ":: CLAKEY$="  ":: QUIZ$="  "
250 CALL SOUND(278,0,2)
260 CALL CLEAR
270 CALL KEY(1,K,ST):: IF ST=0 THEN 270
    ELSE IF K<49 OR K>52 THEN 270
280 IF K=52 THEN 340
290 M$="2^4"
300 CALL SOUND(-4000,0,2)
310 CALL SOUND(-4000,0,2)
320 CALL SOUND(-4000,0,2)
330 CALL SOUND(-4000,0,2)
340 CALL SOUND(-4000,-K*55+440,2)
350 IF ALK=CLK THEN 360 ELSE ALK=CLK::
    CALL KEY(1,K,ST):: IF ST=0 THEN 350
360 CONTINUE
370 IF RIN$="" THEN 390 ELSE RIN$=""::
    CALL SOUND(-4000,0,2)
380 PRINT "YOU NEED TO PUSH 1 OR 2":::
    GOTO 270
390 IF CLAKEY$="1" THEN 400 ELSE IF
    CLAKEY$="2" THEN 410 ELSE 380
400 CALL SOUND(-4000,300,1,110,0)
410 CALL SOUND(-4000,1000,1)
420 NEXT Z
430 PRINT "WORD 1)  WORD 2)":::
440 FOR Z=1 TO 40:PRINT Z;")";Q$(Z,1);
    TAB(15);Q$(Z,2):: NEXT Z
450 INPUT "ENTER NO. OF PAIRS TO QUIZ ON
    (1-40)":NP:: IF NP<1 OR NP>40 THEN 450
460 PRINT "":::
470 PRINT "PRESS 1 FOR WORD 1 OR 2 FOR
    WORD 2":::
480 GOSUB 780
490 IF K=55 THEN 260
500 IF K=54 THEN 570
510 IF TOTG=0 THEN 520 ELSE IF TOTG=NP
    THEN 530
520 RANDOMIZE
530 WSGEV=0
540 FOR K=1 TO NP
550 CALL CLEAR
560 PRINT "QUIZ ";K;"/";NP::
570 CALL CHAR(40,"0000FFFFFFFF")
580 CALL CLEAR
590 CALL CHAR(64,"0000FFFFFFFF")
600 CALL CHAR(96,"0000FFFFFFFF")
610 CALL CHAR(128,"0000FFFFFFFF")
620 CALL CHAR(160,"0000FFFFFFFF")
630 M$="*  *  *  QUIZ-TAKE  *  *  *"
640 XR=18
650 YR=8
660 GOSUB 2410
670 M$="[FCTN] [9] (BACK) TO RETURN"
680 XR=24
690 GOSUB 2400
700 M$=TTL$
710 XR=2
720 YR=INT(16-(LEN(M$)/2))
730 YG=10
740 M$="WR. 0% SP. 0%"
750 XR=3
760 GOSUB 2400
770 SPR=5
780 SXM$=TT$
790 M$=TT$
800 GOSUB 2400
810 M$=TTL$
820 XR=11
830 GOSUB 2400
840 IF QT=1 THEN 870
850 IP=INT(RND*NR)+1
860 IF OTO=IP THEN 850 ELSE OTO=IP
870 IF A=0 THEN 2090
880 IF A=NR THEN 260
890 A=0
900 GOSUB 2090
910 M$=Q$(Z,PROB,1),1,28)
920 XR=7
930 GOSUB 2400
940 IF LEN(Q$(Z,PROB,1))<29 THEN 980
950 M$=Q$(Z,PROB,1),29,28)
960 XR=8
970 GOSUB 2400
980 IF LEV>2 THEN 1000
990 GOSUB 2240
1000 XR=13
1010 YM=27
1020 GOSUB 2450
1030 IF KEY=15 THEN 260
1040 IF K=A$ THEN 1750
1050 ANS$=Q$(PROB,2):ANS$=UCASE$(ANS$)
1060 IF (LEV=1)*(WA=0) THEN 1700
1070 IF (LEV=1)*(WA=1) THEN 1420
1080 IF (LEV=1)*(WA<2) THEN 1300
1090 IF (LEV=3)*(WA=0) THEN 1420
1100 IF (LEV=3)*(WA=2) THEN 1420
1110 IF (LEV=4)*(WA<1) THEN 1300
1120 IF (LEV=4)*(WA=1) THEN 1420
1130 IF (LEV=2)+(LEV=5) THEN 1420
1140 CALL SOUND(500,220,0)
1150 CALL SOUND(700,110,0)
1160 M$="WRONG THE RIGHT ANSWER IS "
1170 XR=15
1180 GOSUB 2400
1190 M$=Q$(PROB,2)
1200 XR=16
1210 GOSUB 2400
1220 FOR TD=1 TO TLIM
1230 CALL SOUND(950,110,30)
1240 CALL SOUND(1,110,10)
1250 NEXT TD
1260 GOSUB 2360
1270 WG=WG+1
1280 TOTG=TOTG+1
1290 GOTO 840
1300 L=LEN(Q$(PROB,2))
1310 CALL SOUND(200,220,0)
1320 CALL SOUND(400,110,0)
1330 NOL=INT(L*LCP/100)
```

Continued

QUIZ-TAKE *Continued*

```
1340 CALL HCHAR(13,1,32,32)
1350 CALL HCHAR(13,3,42,LEN(QZ$(PROB,2))
     )
1360 FOR Z=1 TO NOL
1370 CH=INT(RND*L)+1
1380 CALL HCHAR(13,2+CH,ASC(SEG$(QZ$(PRO
     B,2),CH,1)))
1390 NEXT Z
1400 WA=WA+1
1410 GOTO 1000
1420 PS=0
1430 QZ$(PROB,2)
1440 IF LEN(P$)>LEN(AN$) THEN 1480
1450 AN$=P$
1460 AN$=AN$
1470 GOTO 1500
1480 AN$=P$
1490 FOR Z=1 TO LEN(A$)
1500 IF SEG$(A$,Z,1)=SEG$(B$,Z,1)THEN 15
     30
1510 PC=PC+1
1520 NEXT Z
1530 IF (LEN(P$)-SPC)/LEN(P$)<.7 THEN 11
     40
1550 M$="MISSPELLED-RIGHT SPELLING IS"
1560 GOTO 1500
1570 R=INT(RND*NR)+1
1580 GOTO 2400
1590 RESTORE 2400
1600 GOSUB 2360
1610 RESTORE 2400
1620 GOSUB 2360
1630 FOR Z=1 TO LIMIT
1640 CALL SOUND(950,110,30)
1650 CALL SOUND(1,220,10)
1660 NEXT Z
1670 GOSUB 2360
1680 GOSUB 2370
1690 CALL SOUND(200,220,0)
1700 CALL SOUND(400,110,0)
1710 WA=WA+1
1720 GOTO 1000
1730 FOR Z=1 TO 1000 STEP 50
1740 CALL SOUND(-100,110+Z,0)
1750 NEXT Z
1760 M$="*** THAT IS RIGHT ***"
1770 RESTORE 2400
1780 GOSUB 2360
1790 RESTORE 2400
1800 GOSUB 2360
1810 R=R+1
1820 GOTO 1340
1830 PRINT @8,"ENTER THE DEVICE NAME:"
     ;"AND"
1840 PRINT "FILE NAME FOR DISK:";"CS1
     "
1850 PRINT @13,"FOR OPERATION:"
     OPEN #1:DEV$,INPUT ,FIXED 128,INTER
     NAL
     INPUT #1:TL$,DT$,CN$,QT$
     INPUT #1:AT$,NR,QT,LIM
     PRINT :"READING";NR;"RECORDS";:
     PRINT @13,"PRESS ENTER TO CONTINUE
     "
     FOR Z=1 TO NR
     INPUT #1:QZ$(Z,1),QZ$(Z,2)
     NEXT Z
     PRINT @13,"
     GOSUB 2780
     VOL=1
     GOTO 260
     PRINT @13,"
     PRINT :QZ$(RN,1):::QZ$(RN,2)::"
     GOSUB 2780
     IF (K=10)*(RN<NR)THEN 2050
     IF (K=11)*(RN>1)THEN 2070
     IF K=15 THEN 260
     GOTO 2000
```

```
2050 RN=RN+1
2060 GOTO 1990
2070 RN=RN-1
2080 GOTO 1990
2090 IF OTG=0 THEN 2230
2100 MS=INT(RG/TOTG*100))&"%"
2110 XR=SPOS
2120 YR=7
2130 CALL HCHAR(3,7,32,4)
2140 GOSUB 2410
2150 MS=INT(WG/TOTG*100))&"%"
2160 YR=16
2170 CALL HCHAR(3,16,32,4)
2180 GOSUB 2410
2190 MS=INT(SG/TOTG*100))&"%"
2200 YR=25
2210 CALL HCHAR(3,25,32,4)
2220 GOSUB 2410
2230 RETURN
2240 CALL HCHAR(18,1,32,160)
2250 FOR I=1 TO 5
2260 WC=INT(RND*NR)+1
2270 IF WC=PROB THEN 2260
2280 WC$(Y)=QZ$(WC,2)
2290 NEXT Y
2300 WC$(INT(RND*5)+1)=QZ$(PROB,2)
2310 FOR XR=18 TO 22
2320 M$=WC$(XR-17)
2330 GOSUB 2400
2340 NEXT XR
2350 RETURN
2360 CALL HCHAR(7,1,32,64)
2370 CALL HCHAR(13,1,32,32)
2380 CALL HCHAR(15,1,32,64)
2390 RETURN
2400 YR=3
2410 FOR Z=0 TO LEN(M$)-1
2420 CALL HCHAR(XR,YR+Z,ASC(SEG$(M$,Z+1,
     1)))
2430 NEXT Z
2440 RETURN
2450 A$=""
2460 YR=3
2470 FOR L=0 TO ML-1
2480 CALL HCHAR(XR,YR,CH)
2490 CALL KEY(0,K,S)
2500 CALL HCHAR(XR,YR,95)
2510 CALL HCHAR(XR,YR,CH)
2520 IF S=0 THEN 2490
2530 CALL SOUND(60,110,30,110,30,30000,3
     0,IF (K=13)+(K=15)THEN 2740
2550 IF (YR<30)+(K=7)THEN 2600
2560 CALL HCHAR(XR,YR,K)
2570 YR=3
2580 XR=XR+1
2590 GOTO 2490
2600 CALL HCHAR(XR,YR,K)
2610 IF K<>7 THEN 2710
2620 IF YR=3 THEN 2490
2630 YR=YR-1
2640 GOSUB 2410
2650 A$=SEG$(A$,1,LEN(A$)-1)
2660 IF YR>2 THEN 2690
2670 YR=30
2680 XR=XR-1
2690 CALL HCHAR(XR,YR,CH)
2700 GOTO 2490
2710 A$=A$&CHR$(K)
2720 YR=YR+1
2730 NEXT L
2740 CALL SOUND(80,880,0)
2750 IF LEN(A$)<13 THEN 2770
2760 CALL HCHAR(XR,YR,32)
2770 RETURN
2780 CALL KEY(0,K,S)
2790 IF S=0 THEN 2780
2800 RETURN
2810 PRINT " BYE BYE, SEE YOU NEXT TIME"
2820 END
```

HCM

SKETCH-64

COMMODORE 64

```
100 REM ********KETCH-64*********
110 REM ********BY JAMES P. CHASE
120 REM HOME COMPUTER MAGAZINE
130 REM VOL 4 NO 5 MAY 1986
140 REM CONTAINS 4.5 K
150 REM C64 BASIC
160 REM
170 REM
180 DIM C$(15):DIM T(15):DIM QW(41):DIM
    Q(41):DIM XR(41):DIM YR(41)
190 FOR X=1 TO 15:READ C$(X):NEXT X
200 DATA "{BLK}","{WHT}","{RED}","{CYN}
    ","{PUR}","{GRN}","{BLU}","{YEL}","{
    RVS}","{OFF}","{CLR}","{HOME}","{DOW
    N}","{RIGHT}","{BLK}"
210 FOR X=1 TO 39:READ QW(X):NEXT X
220 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
    0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
    0,0,0,0,0
230 FOR X=1 TO 39:READ Q(X):NEXT X
240 DATA 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
    0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
    0,0,0,0,0
250 PRINT "{CLR}{WHT}":B=53280:S=53281
260 PRINT "{HOME}":CR=54296:RO=40:CX=39
    :RW=24
270 PRINT "{DOWN}":CF=54272:CO=SC+OF:ZZ
    =0
280 PRINT "{DOWN}":POKE B,11:CH=65:CJ=1
    60:CL=5
290 PRINT "{HOME}{DOWN}{CRSR DOWN}{CTRL
    WHT}{TAB}"
300 PRINT "{CRSR DOWN}{CTRL CYN}{RVS}
    {CTRL WHT}{OFF}{CLR}"
310 PRINT "{HOME}{DOWN}{CTRL CYN}{RVS}
    {CTRL WHT}{OFF}{CLR}"
320 PRINT "{HOME}{3 DOWN}{CRSR DOWN}{CT
    RL WHT}{RVS}{OFF}{CLR}"
330 PRINT "{HOME}{3 DOWN}{CRSR DOWN}{CT
    RL WHT}{RVS}{OFF}{CLR}"
340 PRINT "{HOME}{3 DOWN}{CRSR DOWN}{CT
    RL WHT}{RVS}{OFF}{CLR}"
350 PRINT "{HOME}{3 DOWN}{CRSR DOWN}{CT
    RL WHT}{RVS}{OFF}{CLR}"
360 FOR X=1 TO 15:READ T(X):NEXT X
370 DATA 144,5,28,159,156,30,31,158,129
    ,150,151,152,153,154,155
```

Continued

PROGRAM LISTING

```
380 PRINT"{SHIFT CRSRUP}"S$:INPUT"{SHIFT CRSRUP}NEW BORDER COLOR 1{3SHIFT CRSRLEFT}";C$:IFC$=""THENPRINT"{SHIFT CRSRUP}":GOTO 380
390 IFASC(C$)<48ORASC(C$)>57 THEN PRINT"{SHIFT CRSRUP}":GOTO 380
400 C=VAL(C$):IFC<0ORC>15THENPRINT"{SHIFT CRSRUP}":GOTO 380
410 PRINT"{SHIFT CRSRUP}"S$:INPUT"{SHIFT CRSRUP}NEW SCREEN COLOR 0{3SHIFT CRSRLEFT}";D$:IFD$=""THEN PRINT"{SHIFT CRSRUP}":GOTO410
420 IFASC(D$)<48ORASC(D$)>57 THEN PRINT"{SHIFT CRSRUP}":GOTO 410
430 D=VAL(D$):IFD<0ORD>15THENPRINT"{SHIFT CRSRUP}":GOTO 380
440 PRINT"{SHIFT CRSRUP}"S$:INPUT"{SHIFT CRSRUP} ASSIGN A NEW FILE NAME (Y/N) Y{3SHIFT CRSRLEFT}";A$
450 IFA$="N"THEN 470
460 PRINT"{SHIFT CRSRUP}"S$:INPUT"{SHIFT CRSRUP} NAME OF FILE";F$:IF F$=""THEN PRINT"{SHIFT CRSRUP}":GOTO460
470 PRINT"{SHIFT CRSRUP}"S$:INPUT"{SHIFT CRSRUP} FILL SCREEN (Y/N) N{3SHIFT CRSRLEFT}";A$
480 IFA$="N"THENPRINT"{SHIFT CLR}":POKEB,C:POKES,D:GOTO 560
490 PRINT"{SHIFT CRSRUP}"S$:INPUT"{SHIFT CRSRUP} CHAR # (1-255) 83{4SHIFT CRSRLEFT}";T
500 IFT>255THENPRINT"{SHIFT CRSRUP}":GOTO 490
510 PRINT"{SHIFT CRSRUP}"S$:INPUT"{SHIFT CRSRUP} CHAR COLOR (0-15) 1{3SHIFT CRSRLEFT}";O
520 IFO>15THENPRINT"{SHIFT CRSRUP}":GOTO 510
530 PRINT"{SHIFT CLR}":POKEB,C:POKES,D
540 FORI=1024TO2023:POKEI,T:POKE54272+I,O:NEXT
550 POKEB,C:POKES,D
560 POKESC,CH:POKECO,5
570 POKE56320,126
580 REM ** JOYSTICK ROUTINE **
590 POKESC,CJ:POKECO,CL:POKESC,32:POKESC,CH
600 GET A$
610 IFA$="1"THENCL=CL+1:IFCL>15THENCL=0
620 IFA$="2"THENCM=CM+1:GOSUB1060
630 IFA$="3"THENCN=CN+1:GOSUB1090
640 IFA$<>"E"THEN690
650 GOSUB 2080:PRINT "{HOME}{CRSRDOWN}ERASE SCREEN? (Y/N)"
660 GET K$:IF K$="" THEN 660
670 IF K$="Y" THEN PRINT"{SHIFT CLR}":GOTO 690
680 GOSUB 2140
690 IFA$="O"THENZZ=0:IF CH>128THEN CH=CH-128:REM RVS OFF
700 IFA$="S"THENGOSUB1110:REM SAVE
710 IFA$="L"THENGOSUB1710:REM CALCULATE SCREEN
720 IFA$="P"THENGOSUB1820:REM TO PRINTER
730 IFA$="R"THENZZ=1:IF CH<128 THEN CH=CH+128:REM RVS ON
740 IFA$<>"B"THEN790
750 GOSUB 2080:PRINT "{HOME}{CRSRDOWN}RESTART PROGRAM? (Y/N)"
760 GET K$:IF K$="" THEN 760
770 IF K$="Y" THEN RUN
780 GOSUB 2140
790 IFA$<>"Q"THEN840
800 GOSUB 2080:PRINT "{HOME}{CRSRDOWN}QUIT PROGRAM? (Y/N)"
810 GET K$:IF K$="" THEN 810
820 IF K$="Y" THEN SYS 2048
830 GOSUB 2140
840 IFA$="M"THEN1870
850 D=(NOTPEEK(56320))+128
860 IFD=16THENGOSUB 960
870 X=SGN(DAND8)-SGN(DAND4)
880 Y=SGN(DAND2)-SGN(DAND1)
890 IF SC<1064 AND Y=-1 THEN SC=SC+1000
900 IF SC>1983 AND Y=1 THEN SC=SC-1000
910 IFINT((SC-1024)/40)=(SC-1024)/40 AND X=-1 THEN SC=SC+40
920 IFINT(((SC+1)-1024)/40)=((SC+1)-1024)/40 AND X=1 THEN SC=SC-40
930 SC=SC+X+40*Y
940 CO=54272+SC
950 POKESC,CH:POKECO,CL:GOTO 580
960 REM ** PICK CHAR **
970 GOSUB 2080:PRINT "{HOME}{CRSRDOWN}CHOOSE A CHARACTER:";
980 GETA$:IFA$=""THEN 980
990 IF (ASC(A$)>127 AND ASC(A$)<160) OR ASC(A$)<32 THEN 980
1000 PRINTA$:IFASC(A$)>=32THENIFASC(A$)<=63THENCH=ASC(A$):GOTO1040
1010 IFASC(A$)>=64THENIFASC(A$)<=96THENCH=ASC(A$)-64:GOTO1040
1020 IFASC(A$)>=192THENIFASC(A$)<=223THENCH=ASC(A$)-128:GOTO1040
1030 IFASC(A$)>=161THENCH=ASC(A$)-64
1040 IFZZ=1THENCH=CH+128
1050 GOSUB 2140:RETURN
1060 REM ** SCREEN AND BORDER **
1070 POKE53280,CM:IFCM>15THENCM=0
1080 RETURN
1090 POKE53281,CN:IFCN>15THENCN=0
1100 RETURN
1110 REM ** SAVE **
1120 IFF$=""THENGOSUB2010
1130 OPEN5,8,1,"@0:"+F$+",P,W"
1140 GOSUB 2160
1150 IFETHENCLOSE5:CLOSE15:RETURN
1160 F=-1
1170 G%=PEEK(53280)AND15:H%=PEEK(53281)AND15
1180 PRINT#5,G%
1190 PRINT#5,H%
1200 GOSUB1270
1210 M%=PEEK(CR+RX)AND15:N%=PEEK(SR+RX)
1220 PRINT#5,M%
1230 PRINT#5,N%
1240 CLOSE5
1250 CLOSE15
1260 RETURN
1270 REM ** SCREEN CALCULATIONS **
1280 FORY=0TO24
1290 A$="":RF=0:CF=-1
1300 FORX=0TO39
1310 RX=RO*Y+X:M%=PEEK(CR+RX)AND15:N%=PEEK(SR+RX)
1320 IFM%<>CFTHENA$=A$+C$(M%):CF=M%
1330 RS=-(N%AND128)/128
1340 IFRSAND(NOT(RF))THENA$=A$+"{CTRL RVSON}":RF=-1:GOTO1360
1350 IF(NOT(RS))ANDRFTHENA$=A$+"{CTRL RVSOFF}":RF=0
1360 N%=N%AND127
1370 IFN%<32THENN%=N%+64:GOTO1410
1380 IFN%>63ANDN%<96THENN%=N%+128:GOTO1410
1390 IFN%>95THENN%=N%+64
1400 IFN%=32THENN%=160
1410 A$=A$+CHR$(N%)
1420 NEXTX
1430 IFFAND(Y=RW)THENA$=LEFT$(A$,LEN(A$)-1)
1440 IFPP=1THENPRINT#5,CHR$(14)A$CHR$(8):GOTO1460
1450 PRINT#5,CHR$(15)A$CHR$(8)
1460 NEXTY
1470 PP=0
1480 RETURN
1490 REM ** LOAD **
1500 PRINT"{HOME}"
1510 PRINT"{3CRSRDOWN}"S$
1520 INPUT"{SHIFT CRSRUP} FILE NAME";F$
1530 PRINT"{SHIFT CLR}"
1540 ER$="0:"+F$+",P,R"
1550 OPEN5,8,0,ER$
1560 GOSUB 2160
1570 IF E THEN RETURN
1580 INPUT#5,H%:POKE53280,H%
1590 INPUT#5,G%:POKE53281,G%
1600 PRINT"{SHIFT CLR}";
1610 FORY=0TO24
1620 INPUT#5,IN$
1630 PRINTIN$;
1640 NEXTY
1650 INPUT#5,M%,N%:RX=RW*RO+CX
1660 POKESR+RX,N%
1670 POKECR+RX,M%
1680 CLOSE5
1690 CLOSE15
1700 RETURN
1710 REM ** SCREEN CALCULATION **
1720 OPEN4,4
1730 PRINT#4,CHR$(14)TAB(11)"SCREEN POKE VALUES"CHR$(15):PRINT#4
1740 PRINT#4,TAB(40-LEN(F$)/2)F$:PRINT#4
1750 FORI=1TO80:PRINT#4,"{CMDR T}";:NEXT
1760 SC=1023:CR=55295
1770 FORI=1TO1000:LL=PEEK(SC+I):MM=PEEK(CR+I)AND15
1780 PRINT#4,SC+I;LL;MM;TAB(4):DS=DS+1:IFDS=4THENPRINT#4:DS=0
1790 NEXT
1800 CLOSE4
1810 RETURN
1820 OPEN5,4
1830 PP=1
1840 GOSUB1270
1850 CLOSE5
1860 RETURN
1870 REM ** CROSS SCREEN **
1880 GETA$:IFA$="M"THEN 560
1890 V=PEEK(56320)AND15
1900 C=T(V)
1910 SC=SC+C:CO=CO+C
1920 P=PEEK(SC):PP=PEEK(CO)
1930 IFSC>=2023THENSC=2023
1940 IFSC<=1024THENSC=1024
1950 IFCO>=56295THENCO=56295
1960 IFCO<=55296THENCO=55296
1970 POKESC,160:FORI=1TO25:NEXT:POKESC,32
1980 POKESC,P
1990 GOTO1880
2000 FORI=WTOW+39:POKEI,WQ(F):F=F+1:NEXT:F=0
```

Continued

SKETCH-64 *Continued*

COMMODORE 64

```
2010 REM ** NO FILE NAME **
2020 GOSUB 2080
2030 PRINT"{HOME}{CRSR DOWN} NO FILE NAME"
2040 FORI=1TO1500:NEXT
2050 PRINT"{HOME}{CRSR DOWN}";:INPUT"FILE NAME PLEASE:";F$:IF F$="" THEN GOSUB 2080:GOTO 2050
2060 GOSUB 2140
2070 RETURN
2080 Q=1064:W=55336:IF S<>1 THEN PRINT"{CTRL WHT}";:GOTO 2100
2090 PRINT "{CTRL BLK}";
2100 J=0:FORI=QTOQ+39:QW(J)=PEEK(I):J=J+1:NEXT:J=0
2110 F=0:FORI=WTOW+39:WQ(F)=PEEK(I):F=F+1:NEXT:F=0
2120 G=PEEK(646):H=PEEK(53281)AND15:IFG=HTHENPRINT"{CTRL BLK}";
2130 FORI=QTOQ+39:POKEI,32:NEXT:RETURN
2140 J=0:FORI=QTOQ+39:POKEI,QW(J):J=J+1:NEXT:J=0
2150 FORI=WTOW+39:POKEI,WQ(F):F=F+1:NEXT:F=0:RETURN
2160 OPEN15,8,15
2170 INPUT#15,E,E$,E1,E2
2180 IFETHENGOSUB2080:PRINT"{HOME}{CRSR DOWN}";E$;", TRY AGAIN":FOR I=1TO1500:NEXT:GOSUB2140
2190 RETURN
```

HCM

SLITHER

APPLE // Family

```
100 REM ***********
110 REM * SLITHER *
120 REM ***********
130 REM BY AARON CHEW
140 REM AND THE HCM STAFF
150 REM HOME COMPUTER MAGAZINE
160 REM VERSION 4.5.1
170 REM APPLE II FAMILY APPLESOFT
180 IF PEEK (103) + PEEK (104) * 256 < > 24577 THEN POKE 24576,0: POKE 103,24577 - INT (24577 / 256) * 256: POKE 104, INT (24577 / 256): PRINT CHR$ (4);"RUN SLITHER"
190 CLEAR
200 REM ******************
210 REM ** SHAPE TABLES **
220 REM ******************
230 FOR I = 16384 TO 16384 + 83: READ V: POKE I,V: NEXT
240 DATA 6,0,14,0,25,0,35,0,46,0,63,0,74,0,45,45,62,63,55,45,45,62,63,63,0,41,53,53,55,63,60,44,53,6,0
250 DATA 45,45,254,27,46,45,53,63,63,7,0,45,45,54,62,63,36,55,54,0,45,54,62,63,36,55,54,0,45,45,62,63,55,77,49,63,63,7,0,45,45,54,62,36,151,63,36,4,0
260 POKE 232,0: POKE 233,64: SCALE= 1: ROT= 0: HCOLOR= 3
270 GOSUB 1330
280 REM *****************
290 REM * SOUND ROUTINE *
300 REM *****************
310 FOR I = 771 TO 789: READ V: POKE I,V: NEXT
320 DATA 173,48,192,136,208,4,198,1,240,8,202,208,246,166,0,76,3,3,96
330 REM *****************
340 REM * INIT VARIABLES *
350 REM *****************
360 MZ = 1:SC = 0:LI = 3:NL = 500: DIM SN%(2240,1)
370 X = 7:Y = 2:N = 1:L = 1:C = 5:DI = 2:FIRST = 1
380 REM ***************
390 REM * INIT SCREEN *
400 REM ***************
410 HOME : GR : HGR : COLOR= 9: FOR I = 0 TO 39: HLIN 0,39 AT I: NEXT
420 CALL - 3086
430 HCOLOR= 6: FOR I = 0 TO 55: DRAW 1 AT (I * 5),0: DRAW 1 AT (I * 5),156: NEXT
440 FOR I = 0 TO 39: DRAW 1 AT 0,(I * 4): DRAW 1 AT (55 * 5),(I * 4): NEXT
450 IF MZ > 1 THEN HCOLOR= 1: FOR I = 3 TO 52: DRAW 1 AT (I * 5),(3 * 4): DRAW 1 AT (I * 5),(36 * 4): NEXT : FOR I = 3 TO 36: DRAW 1 AT (3 * 5),(I * 4): DRAW 1 AT (52 * 5),(I * 4): NEXT :RO = 3: GOSUB 510
460 IF MZ > 2 THEN HCOLOR= 2: FOR I = 6 TO 49: DRAW 1 AT (I * 5),(6 * 4): DRAW 1 AT (I * 5),(33 * 4): NEXT : FOR I = 6 TO 33: DRAW 1 AT (6 * 5),(I * 4): DRAW 1 AT (49 * 5),(I * 4): NEXT :RO = 33: GOSUB 510
470 IF MZ > 3 THEN HCOLOR= 5: FOR I = 9 TO 46: DRAW 1 AT (I * 5),(9 * 4): DRAW 1 AT (I * 5),(30 * 4): NEXT : FOR I = 9 TO 30: DRAW 1 AT (9 * 5),(I * 4): DRAW 1 AT (46 * 5),(I * 4): NEXT :RO = 9: GOSUB 510
480 IF MZ > 4 THEN HCOLOR= 7: FOR I = 12 TO 43: DRAW 1 AT (I * 5),(12 * 4): DRAW 1 AT (I * 5),(27 * 4): NEXT : FOR I = 12 TO 27: DRAW 1 AT (12 * 5),(I * 4): DRAW 1 AT (43 * 5),(I * 4): NEXT :RO = 27: GOSUB 510
490 IF MZ > 5 THEN HCOLOR= 3: FOR I = 15 TO 40: DRAW 1 AT (I * 5),(15 * 4): DRAW 1 AT (I * 5),(24 * 4): NEXT : FOR I = 15 TO 24: DRAW 1 AT (15 * 5),(I * 4): DRAW 1 AT (40 * 5),(I * 4): NEXT :RO = 15: GOSUB 510
500 GOTO 550
510 HCOLOR= 0: FOR I = 26 TO 29: DRAW 1 AT (I * 5),(RO * 4): NEXT : RETURN
520 REM *****************
530 REM * EGG POSITION *
540 REM *****************
550 X2 = INT ( RND ( PEEK (78) + PEEK (79) * 256) * 56):Y2 = INT ( RND ( PEEK (78) + PEEK (79) * 256) * 40)
560 XDRAW 2 AT (X2 * 5),(Y2 * 4):CL = PEEK (234): XDRAW 2 AT (X2 * 5),(Y2 * 4)
570 IF CL < > 16 THEN 500
580 HCOLOR= 3: DRAW 2 AT (X2 * 5),(Y2 * 4)
590 REM *********************
600 REM * MAIN PROGRAM LOOP *
610 REM *********************
620 KB = PEEK ( - 16384): POKE - 16368,0
630 IF FIRST = 1 THEN FIRST = 0: GOTO 650
640 HCOLOR= 3: DRAW 1 AT (X * 5),(Y * 4)
650 IF KB = 199 THEN DI = DI - 1: IF DI < 1 THEN DI = 4
660 IF KB = 200 THEN DI = DI + 1: IF DI > 4 THEN DI = 1
670 VTAB 22: HTAB 1: PRINT "SCORE: ";SC;: SPC( 10);"LIVES: ";LI:Q = FRE (0)
680 ON DI GOTO 700,710,720,730
690 GOTO 740
700 Y = Y - 1: GOTO 740
710 X = X + 1: GOTO 740
720 Y = Y + 1: GOTO 740
730 X = X - 1
740 SN%(N,0) = X:SN%(N,1) = Y:N = N + 1: IF N = 2240 THEN N = 1
750 REM ********************
760 REM *CHECK FOR OBSTACLES*
770 REM ********************
780 XDRAW 1 AT (X * 5),(Y * 4):CL = PEEK (234): XDRAW 1 AT (X * 5),(Y * 4)
790 IF CL < > 4 AND CL < > 20 THEN 1200
800 IF CL = 4 THEN FOR I = 65 TO 50 STEP - 1: POKE 0,I: POKE 1,3: CALL 771: NEXT
810 IF CL = 4 THEN RP = 1:SC = SC + 20:C = C + 15: VTAB 22: PRINT "SCORE: ";SC
820 SH = DI + 2
830 HCOLOR= 3: DRAW SH AT (X * 5),(Y * 4)
840 IF C = 0 THEN XDRAW 1 AT (SN%(L,0) * 5),(SN%(L,1) * 4):L = L + 1: IF L = 2240 THEN L = 1
850 IF C > 0 THEN C = C - 1
860 IF SC = NL THEN 1060
870 IF RP = 1 THEN RP = 0: GOTO 500
880 MV = INT ( RND (1) * 25) + 1: IF MV = 1 THEN 930
890 GOTO 620
900 REM *********************
910 REM *CHANGE EGG POSITION*
920 REM *********************
930 MV = INT ( RND (1) * 4) + 1
940 ON MV GOTO 950,970,990,1010
950 Y2 = Y2 - 1: XDRAW 2 AT (X2 * 5),(Y2 * 4):CL = PEEK (234): IF CL = 16 THEN XDRAW 2 AT (X2 * 5),((Y2 + 1) * 4): GOTO 620
960 XDRAW 2 AT (X2 * 5),(Y2 * 4):Y2 = Y2 + 1: GOTO 620
970 X2 = X2 + 1: XDRAW 2 AT (X2 * 5),(Y2 * 4):CL = PEEK (234): IF CL = 16 THEN XDRAW 2 AT ((X2 - 1) * 5),(Y2 * 4): GOTO 620
980 XDRAW 2 AT (X2 * 5),(Y2 * 4):X2 = X2 - 1: GOTO 620
990 Y2 = Y2 + 1: XDRAW 2 AT (X2 * 5),(Y2 * 4):CL = PEEK (234): IF CL = 16 THEN XDRAW 2 AT (X2 * 5),((Y2 - 1) * 4): GOTO 620
```

Continued

PROGRAM LISTING

SLITHER *Continued*

APPLE // Family

```
1000 XDRAW 2 AT (X2 * 5),(Y2 * 4):Y2 = Y2 - 1: GOTO 620
1010 X2 = X2 - 1: XDRAW 2 AT (X2 * 5),(Y2 * 4):CL = PEEK (234): IF CL = 16 THEN XDRAW 2 AT ((X2 + 1) * 5),(Y2 * 4): GOTO 620
1020 XDRAW 2 AT (X2 * 5),(Y2 * 4):X2 = X2 + 1: GOTO 620
1030 REM *********************
1040 REM * INTERMISSION TIME *
1050 REM *********************
1060 TEXT : HOME
1070 VTAB 11: PRINT "CONGRATULATIONS!": PRINT : PRINT "YOU HAVE JUST COMPLETED MAZE ";MZ;".": PRINT "YOU ARE AWARDED AN EXTRA SLITHER."
1080 FOR J = 1 TO 2: FOR I = 1 TO 2: POKE 0,120: POKE 1,100: CALL 771: NEXT : POKE 1,140: CALL 771
1090 IF J = 2 THEN 1120
1100 FOR T = 1 TO 200: NEXT
1110 NEXT
1120 READ A,B: IF A = 0 THEN 1150
1130 POKE 0,A: POKE 1,B: CALL 771: GOTO 1120
1140 DATA 100,100,112,100,120,100,134,100,149,100,100,90,149,90,0,0
1150 RESTORE : FOR I = 1 TO 103: READ A: NEXT : PRINT : PRINT "PRESS <RETURN> TO CONTINUE";: CALL - 678
1160 RP = 0:MZ = MZ + 1:NL = NL + 500:LI = LI + 1: GOTO 370
1170 REM ************
1180 REM ** CRASH! **
1190 REM ************
1200 FOR I = 1 TO 25
1210 POKE - 16298,0: POKE 0,I * 10: POKE 1,5: CALL 771: POKE - 16297,0: POKE 0,I: POKE 1,5: CALL 771: NEXT
1220 FOR T = 1 TO 1000: NEXT :LI = LI - 1: IF LI = 0 THEN 1270
1230 POKE - 16368,0: GOTO 370
1240 REM ***************
1250 REM * END OF GAME *
1260 REM ***************
1270 TEXT : HOME : POKE - 16368,0: PRINT : PRINT "YOUR SCORE IS: ";SC: PRINT : PRINT "DO YOU WANT TO PLAY AGAIN";: INPUT A$
1280 IF LEFT$ (A$,1) = "Y" THEN RUN
1290 END
1300 REM ****************
1310 REM * INTRODUCTION *
1320 REM ****************
1330 TEXT : HOME :T$ = "S L I T H E R"
1340 VTAB 16: HTAB 9: PRINT "PRESS <RETURN> TO PLAY"
1350 VTAB 12: HTAB 14: PRINT T$: FOR T = 1 TO 400: NEXT
1360 VTAB 12: HTAB 14: PRINT "              "
1370 KB = PEEK ( - 16384): IF KB = 141 THEN POKE - 16368,0: RETURN
1380 FOR T = 1 TO 400: NEXT : GOTO 1350
```

HCM

SLITHER

COMMODORE 64

```
100 REM **********
110 REM * SLITHER *
120 REM **********
130 REM BY AARON CHEW
140 REM AND THE HCM STAFF
150 REM HOME COMPUTER MAGAZINE
160 REM VERSION 4.5.1
170 REM C-64 BASIC
180 REM
190 GOSUB 1600
200 DIMC(17),E(31),L(57),B(960),T(960)
210 FORW=1TO16:READQ1:C(W)=Q1:NEXTW
220 FORW=1TO56:READQ2:L(W)=Q2:NEXTW
230 FORW=1TO30:READQ3:E(W)=Q3:NEXTW
240 GOSUB 1620
250 MAZE=1:SC=0:LI=3:NXSC=500:MT=0:AD=1064:Q=54272:TZ=5
260 PRINT"{SHIFT CLR}":POKE53281,15:POKE53280,15
270 GOSUB 480::GOSUB 740
280 IFCH=75THENGOSUB 1200::GOTO 300
290 IF CH=74THENGOSUB 1150
300 FOR XX=1TO20 :NEXTXX
310 ONNWDIRGOTO 320, 330, 340, 350
320 HH=H-1:DIR=1:GOTO 360
330 HH=H-40:DIR=2:GOTO 360
340 HH=H+1:DIR=3:GOTO 360
350 HH=H+40:DIR=4:GOTO 360
360 QD=B(HH-AD)
370 IFQD=1THENGOSUB 1280::LI=LI-1
380 IFQD=1THENFORX=LTTOEZ-1:S=T(X):GOSUB 1140::T(X)=0:NEXTX:IFLI=0THEN 990
390 IFQD=1THENGOSUB 1090::S=T:GOSUB 1140::GOSUB 710::GOSUB 740::TZ=5:GOTO 280
400 IFQD=2THENB=H:H=HH:GOSUB 1120::GOSUB 1110::GOSUB 1460::SC=SC+20
410 IFQD=2THENGOSUB 1080::IFSC=NXSCTHEN 870
420 IFQD=2THENTZ=TZ+5:GOSUB 740::GOTO 440
430 B=H:H=HH:GOSUB 1120::GOSUB 1110
440 EX=(EZ-LT)-TZ
450 IFEX>0THENS=T(LT):GOSUB 1140::T(LT)=0:LT=LT+1
460 MT=MT+1:IFMT=5THENGOSUB 770::MT=0
470 GOTO 280
480 FORW=1064TO1104:GOSUB 1100::NEXTW
490 FORW=1143 TO1983STEP40:GOSUB 1100::NEXTW
500 FORW=1144TO1984STEP40:GOSUB 1100::NEXTW
510 FORW=1984TO2023:GOSUB 1100::NEXTW
520 GOSUB 1080::GOSUB 1090
530 IF MAZE<2GOTO 710
540 FORW=1187TO1201:GOSUB 1100::NEXTW
550 FORW=1206TO1220:GOSUB 1100::NEXTW
560 FORW=1227TO1827STEP40:GOSUB 1100::NEXTW
570 FORW=1260TO1860STEP40:GOSUB 1100::NEXTW
580 FORW=1867TO1900:GOSUB 1100::NEXTW
590 IF MAZE<3GOTO 710
600 FORW=1310TO1337:GOSUB 1100::NEXTW
610 FORW=1350TO1710STEP40:GOSUB 1100::NEXTW
620 FORW=1377TO1737STEP40:GOSUB 1100::NEXTW
630 FORW=1750TO1761:GOSUB 1100::NEXTW
640 FORW=1766TO1777:GOSUB 1100::NEXTW
650 IFMAZE<4GOTO 710
660 FORW=1433TO1441:GOSUB 1100::NEXTW
670 FORW=1446TO1454:GOSUB 1100::NEXTW
680 FORW=1473TO1593STEP40:GOSUB 1100::NEXTW
690 FORW=1494TO1614STEP40:GOSUB 1100::NEXTW
700 FORW=1633TO1654:GOSUB 1100::NEXTW
710 H=1113:GOSUB 1120::DIR=3:T(5)=1113
720 FORB=1112TO1109STEP-1:GOSUB 1110::NEXTB
730 FORX=0TO4:T(X)=1109+X:NEXTX:EZ=5:LT=0:RETURN
740 T=INT(RND(1)*(877)+1105)
750 IFB(T-AD)<>0GOTO 740
760 GOSUB 1130::MT=0:RETURN
770 MN=INT(8*RND(1))+1
780 IFINT(MN/2)*2=MNTHENRETURN
790 IFMN=1THENTH=T-1:GOTO 830
800 IFMN=3THENTH=T-40:GOTO 830
810 IFMN=5THENTH=T+1:GOTO 830
820 IF MN=7THENTH=T+40
830 IFTH<1065ORTH>1982THENRETURN
840 IFB(TH-AD)<>0THENTH=0:RETURN
850 S=T:GOSUB 1140
860 T=TH:GOSUB 1130::RETURN
870 REM NEXT LEVEL
880 NXSC=NXSC+500:LIVES=LIVES+1
890 FORX=LTTOEZ-1:S=T(X):GOSUB 1140::T(X)=0:NEXTX
900 PRINT"{SHIFT CLR}":POKE53281,3:POKE53280,3
910 PRINT"{2 CRSR RIGHT}{5 CRSR DOWN}{CTRL BLK}CONGRATULATIONS!!!!!"
920 PRINT"YOU HAVE JUST COMPLETED MAZE"STR$(MAZE)"."
930 PRINT"{CRSR DOWN}YOU ARE AWARDED AN EXTRA SLITHER.":GOSUB 1370
940 PRINT"{CRSR DOWN}PLEASE PRESS THE 'C' KEY TO CONTINUE."
950 K$="":GETK$
960 IFK$<>"C"GOTO 950
970 MAZE=MAZE+1:PRINT"{SHIFT CLR}":TZ=5
980 GOTO 260
990 PRINT"{SHIFT CLR}":POKE53281,3:POKE53280,0
1000 PRINT"{HOME}{CTRL BLK}{5 CRSR RIGHT}{10 CRSR DOWN}YOUR SCORE IS:"SC
1010 PRINT"{3 CRSR DOWN}DO YOU WANT TO PLAY AGAIN (Y/N)?"
1020 K$="":GETK$:IFK$=""THEN 1020
1030 IFK$="N"THENPOKE53272,21:PRINT"{SHIFT CLR}":END
1040 IFK$<>"Y"THEN 1020
1050 PRINT"{2 CRSR DOWN}ONE MOMENT PLEASE WHILE WE GET READY"
1060 FORX=0TO959:B(X)=0:NEXTX
1070 PRINT"{SHIFT CLR}":GOTO 250
1080 PRINT"{HOME}{CTRL WHT}SCORE:"SC:RETURN
1090 PRINT"{HOME}{15 CRSR RIGHT}{CTRL WHT}LIVES:"LI:RETURN
1100 POKEW,102:POKEW+Q,0:B(W-AD)=1:RETURN
1110 POKEB,81:POKEB+Q,6:B(B-AD)=1:RETURN
1120 POKEH,81:POKEH+Q,2:B(H-AD)=1:T(EZ)=H:EZ=EZ+1:RETURN
1130 POKET,90:POKET+Q,1:B(T-AD)=2:RETURN
1140 POKES,32:POKES+Q,15:B(S-AD)=0:RETURN
1150 HI=((PEEK(56321)AND 4)=0)-((PEEK(56321)AND 8)=0)
1160 IFHI=0THENNWDIR=DIR
```

Continued

```
1170 IFHI=1THENNWDIR=DIR+1:IFNWDIR=5THENNWDIR=1
1180 IFHI=-1THENNWDIR=DIR-1:IFNWDIR=0THENNWDIR=4
1190 RETURN
1200 REM KEYBOARD
1210 GETRI$
1220 IFRI$="H"THENNWDIR=DIR+1:IFNWDIR=5THENNWDIR=1
1230 IFRI$="G"THENNWDIR=DIR-1:IFNWDIR=0THENNWDIR=4
1240 IFRI$=""THENRY=RY+1
1250 IF RY<20THENGOTO 1210
1260 IFRI$=""THENNWDIR=DIR
1270 RETURN
1280 REM MUSIC FOR CRASH
1290 FORS=54272TO54296:POKES,0:NEXTS
1300 FORY=1TO15STEP2:SH=C(Y):SL=C(Y+1)
1310 IFSH=-1THEN 1440
1320 POKE54273,SH:POKE54272,SL
1330 POKE54277,100:POKE54278,100:POKE54276,33:POKE54296,15
1340 FORX=1TO60:NEXTX:NEXTY
1350 POKE54296,0:POKE54276,32
1360 RETURN
1370 REM MUSIC FOR NEW LEVEL
1380 FORS=54272TO54296:POKES,0:NEXTS:Y=1
1390 FORY=1TO55STEP2:SH=L(Y):SL=L(Y+1)
1400 IFSH=-1THEN 1440
1410 POKE54273,SH:POKE54272,SL
1420 POKE54277,100:POKE54278,100:POKE54276,33:POKE54296,15
1430 FORX=1TO25:NEXTX:NEXTY
1440 POKE54296,0:POKE54276,32
1450 RETURN
1460 REM MUSIC FOR GETTING EGG
1470 FORS=54272TO54296:POKES,0:NEXTS:Y=1
1480 FORY=1TO29STEP2:SH=E(Y):SL=E(Y+1)
1490 IFSH=-1THEN 1530
1500 POKE54273,SH:POKE54272,SL
1510 POKE54277,100:POKE54278,100:POKE54276,33:POKE54296,15
1520 FORX=1TO45:NEXTX:NEXTY
1530 POKE54296,0:POKE54276,32
1540 RETURN
1550 DATA28,49,25,30,22,96,21,31,18,209,16,195,15,210,-1,-1
1560 DATA25,30,0,0,25,30,0,0,21,31,0,0,25,30,0,0,28,49,0,0,25,30,0,0,21,31,0,0
1570 DATA0,0,0,0,0,0,21,31,0,0,18,209,0,0,0,0,0,0,0,21,31,0,0,18,209,-1,-1
1580 DATA16,195,0,0,16,195,0,0,22,96,0,0,22,96,0,0,22,96,21,31,0,0,25,30
1590 DATA0,0,25,30,-1,-1
1600 PRINT"{SHIFT CLR}{CTRL WHT}{10 CRSR DOWN}{14 CRSR RIGHT}S L I T H E R"
1610 PRINT"{2 CRSR DOWN}":RETURN
1620 PRINT"{SHIFT CLR}{6 CRSR DOWN}DO YOU WANT TO USE JOYSTICK OR KEYBOARD?"
1630 PRINT"{4 CRSR RIGHT}(KEY 'G' FOR LEFT)"
1640 PRINT"{4 CRSR RIGHT}(KEY 'H' FOR RIGHT)"
1650 PRINT"{3 CRSR DOWN}PRESS 'J' FOR JOYSTICK IN PORT #1"
1660 PRINT"{CRSR DOWN}PRESS 'K' FOR KEYBOARD"
1670 GETP$:IFP$=""THEN 1670
1680 IFP$="J"THENCH=74:RETURN
1690 IFP$="K"THENCH=75:RETURN
1700 GOTO 1660
```

HCM

SLITHER

IBM PC & IBM PCjr

```
100 ' ***********
110 ' * SLITHER *
120 ' ***********
130 ' BY AARON CHEW
140 ' HOME COMPUTER MAGAZINE
150 ' VERSION 4.5.1
160 ' IBM PCjr CARTRIDGE BASIC OR
170 ' IBM PC BASICA WITH
180 ' COLOR/GRAPHICS MONITOR ADAPTER
190 ' AND COLOR MONITOR
200 OPTION BASE 1
210 DIM SNAKE%(2356,2),SC(21),N$(21),DIR(4,2)
220 DATA 0,-1,1,0,0,1,-1,0
230 FOR X=1 TO 4:READ DIR(X,1),DIR(X,2):NEXT X
240 '*** Graphic characters ***
250 KEY OFF:GOSUB 1280
260 SCREEN 1:COLOR 0,0:CLS
270 DIM HEAD1%(7),HEAD2%(7),HEAD3%(7),HEAD4%(7),BLOCK%(7),TARGET%(7),CRASH%(7)
280 DATA 22322,23232,31213,31213,32223
290 FOR X=1 TO 5:READ X$:FOR X1=1 TO 5:PSET(X1-1,X-1),VAL(MID$(X$,X1,1)):NEXT X1:NEXT X:GET(0,0)-(4,4),HEAD1%
300 DATA 01020,00103,00200,10020,01030
310 FOR X=1 TO 5:READ X$:FOR X1=1 TO 5:PSET(X1-1,X-1),VAL(MID$(X$,X1,1)):NEXT X1:NEXT X:GET(0,0)-(4,4),CRASH%
320 DATA 33322,21132,22223,21132,33322
330 FOR X=1 TO 5:READ X$:FOR X1=1 TO 5:PSET(X1-1,X-1),VAL(MID$(X$,X1,1)):NEXT X1:NEXT X:GET(0,0)-(4,4),HEAD2%
340 DATA 32223,31213,31213,23232,22322
350 FOR X=1 TO 5:READ X$:FOR X1=1 TO 5:PSET(X1-1,X-1),VAL(MID$(X$,X1,1)):NEXT X1:NEXT X:GET (0,0)-(4,4),HEAD3%
360 DATA 22333,23112,32222,23112,22333
370 FOR X=1 TO 5:READ X$:FOR X1=1 TO 5:PSET(X1-1,X-1),VAL(MID$(X$,X1,1)):NEXT X1:NEXT X:GET (0,0)-(4,4),HEAD4%
380 LINE(0,0)-(4,4),2,BF
390 GET(0,0)-(4,4),BLOCK%
400 DATA 11111,13031,10301,13031,11111
410 FOR X=1 TO 5:READ X$:FOR X1=1 TO 5:PSET(X1-1,X-1),VAL(MID$(X$,X1,1)):NEXT X1:NEXT X:GET (0,0)-(4,4),TARGET%
420 '*** Initialize Screen ***
430 LINE (0,0)-(319,4),3,BF:LINE(315,0)-(319,189),3,BF
440 LINE (0,0)-(4,189),3,BF:LINE(0,185)-(319,189),3,BF
450 FOR X=5 TO 20 STEP 5:PUT(X,5),BLOCK%,XOR:NEXT X
460 DEF SEG: POKE &H4E,1:LOCATE 25,1:PRINT "SCORE:";:LOCATE 25,20:PRINT "LIVES:";:DEF SEG: POKE &H4E,2:IF TWICE=0 THEN LOCATE 25,7:PRINT 0;:LOCATE 25,26:PRINT 3; ELSE LOCATE 25,7:PRINT SCORE;:LOCATE 25,26:PRINT LIVES;
470 IF MAZE>1 THEN LINE(15,15)-(149,19),3,BF:LINE(170,15)-(304,19),3,BF:LINE(15,15)-(19,170),3,BF:LINE(15,170)-(304,174),3,BF:LINE(300,15)-(304,170),3,BF
480 IF MAZE>2 THEN LINE(30,30)-(289,34),3,BF:LINE(30,30)-(34,156),3,BF:LINE(30,155)-(149,159),3,BF:LINE (170,155)-(289,159),3,BF:LINE(285,30)-(289,155),3,BF
490 IF MAZE>3 THEN LINE(45,45)-(149,49),3,BF:LINE(170,45)-(274,49),3,BF:LINE(45,45)-(49,140),3,BF:LINE(45,140)-(274,144),3,BF:LINE(270,45)-(274,140),3,BF
500 IF MAZE>4 THEN LINE(60,60)-(259,64),3,BF:LINE(60,60)-(64,126),3,BF:LINE(60,125)-(149,129),3,BF:LINE (170,125)-(259,129),3,BF:LINE(255,60)-(259,125),3,BF
510 IF MAZE>5 THEN LINE(75,75)-(149,79),3,BF:LINE(170,75)-(244,79),3,BF:LINE(75,75)-(79,110),3,BF:LINE(75,110)-(244,114),3,BF:LINE(240,75)-(244,110),3,BF
520 '*** Initialize variables ***
530 IF TWICE=1 THEN 570 ELSE TWICE=1
540 CHANGEM=50
550 P$="a:slither":F$="a:fslither":T$=" S L I T H E R ":G=2170
560 SCORE=0:SCOREC=500:MAZE=1:LIVES=3
570 FOR X=1 TO 5:SNAKE%(X,1)=5:NEXT X
580 FOR X=1 TO 5:SNAKE%(X,2)=X*5:NEXT X
590 BPOINTER=0:FPOINTER=5:ACTIVE=0:XWAIT=0:XWAITI=0:CHECK=0:DIRC=1:DIRR=0:DP=2
600 ROWTAR=(INT(RND*37)+1)*5:COLTAR=(INT(RND*63)+1)*5:IF POINT(COLTAR,ROWTAR)<>0 THEN 600 ELSE PUT(COLTAR,ROWTAR),TARGET%,XOR
610 ROW=5:COL=25:CHANGEI=0:CHANGE=1
620 '*** Main program loop
630 DEF SEG=0:POKE 1050,PEEK(1052)
640 K$=INKEY$:IF K$="" THEN 730
650 IF K$="8" THEN DIRR=-1:DIRC=0:DP=1:GOTO 730
660 IF K$="6" THEN DIRC=1:DIRR=0:DP=2:GOTO 730
670 IF K$="2" THEN DIRR=1:DIRC=0:DP=3:GOTO 730
680 IF K$="4" THEN DIRC=-1:DIRR=0:DP=4:GOTO 730
690 IF K$<>"g" AND K$<>"G" AND K$<>"h" AND K$<>"H" THEN 730
700 IF K$="g" OR K$="G" THEN DP=DP-1:IF DP=0 THEN DP=4
710 IF K$="h" OR K$="H" THEN DP=DP+1: IF DP=5 THEN DP=1
720 DIRC=DIR(DP,1):DIRR=DIR(DP,2)
730 ROW=ROW+DIRR*5:COL=COL+DIRC*5
740 CHANGEI=CHANGEI+1:IF CHANGEI>CHANGE THEN GOSUB 970
750 CHECK=POINT(COL,ROW)
760 IF CHECK<>0 THEN 890
770 IF DIRR=-1 THEN PUT(COL,ROW),HEAD1%,PSET
```

Continued

SLITHER *Continued*

IBM PC & IBM PCjr

```
 780 IF DIRC=1 THEN PUT(COL,ROW),HEAD2%,
     PSET
 790 IF DIRR=1 THEN PUT(COL,ROW),HEAD3%,
     PSET
 800 IF DIRC=-1 THEN PUT(COL,ROW),HEAD4%
     ,PSET
 810 PUT(COL-DIRC*5,ROW-DIRR*5),BLOCK%,P
     PSET
 820 FPOINTER=FPOINTER+1:IF FPOINTER=235
     7 THEN FPOINTER=1
 830 SNAKE%(FPOINTER,1)=ROW:SNAKE%(FPOIN
     TER,2)=COL
 840 IF ACTIVE=1 THEN XWAITI=XWAITI+1:IF
      XWAITI>XWAIT THEN ACTIVE=0:XWAITI=
     0:XWAITI=0
 850 IF ACTIVE=1 THEN 640
 860 BPOINTER=BPOINTER+1:IF BPOINTER=235
     7 THEN BPOINTER=1
 870 PUT((SNAKE%(BPOINTER,2),SNAKE%(BPOIN
     TER,1)),BLOCK%,XOR:GOTO 640
 880 '*** Check for obstacles ***
 890 IF CHECK<>1 THEN 930
 900 PUT((COLTAR,ROWTAR),TARGET%,XOR:FOR
     X=0 TO 5 STEP 2:PLAY"mbt255164o=x;a
     cdefg:NEXT X:ACTIVE=1: XWAIT=XWAI
     T+15 :SCORE=SCORE+20:LOCATE 25,7:PR
     INT SCORE;
 910 IF SCORE>=SCOREC THEN GOSUB 1150:CL
     S:GOTO 430
 920 COLTAR=(INT(RND*63)+1)*5:ROWTAR=(IN
     T(RND*37)+1)*5:IF POINT(COLTAR,ROWT
     AR)<>0 THEN 920 ELSE PUT(COLTAR,ROW
     TAR),TARGET%,XOR:GOTO 770
 930 PUT((COL-DIRC*5,ROW-DIRR*5),,BLOCK%,P
     SET:PUT((COL,ROW),CRASH%,PSET:PLAY"
     mbt1201404n1":FOR X=1 TO 200:NEXT X:P
     UT(COL,ROW),CRASH%,XOR
 940 IF LIVES=LIVES-1:IF LIVES=0 THEN 1440
 950 CLS:GOTO 430
 960 '*** Change target ***
 970 CHANGEI=0:CHANGEI=INT((CHANGEM*.75*RN
     D)+INT(CHANGEM*.25)):ON INT(RND*4+1)
     GOTO 980,1020,1060,1100
 980 IF POINT(COLTAR,ROWTAR-5)=0 THEN PU
     T(COLTAR,ROWTAR),TARGET%,XOR:ROWTAR
     =ROWTAR-5:PUT(COLTAR,ROWTAR),TARGET
     %,XOR:RETURN
 990 IF POINT(COLTAR,ROWTAR+5)=0 THEN PU
     T(COLTAR,ROWTAR),TARGET%,XOR:ROWTAR
     =ROWTAR+5:PUT(COLTAR,ROWTAR),TARGET
     %,XOR:RETURN
1000 IF POINT(COLTAR-5,ROWTAR)=0 THEN PU
     T(COLTAR,ROWTAR),TARGET%,XOR:COLTAR
     =COLTAR-5:PUT(COLTAR,ROWTAR),TARGET
     %,XOR:RETURN
1010 IF POINT(COLTAR+5,ROWTAR)=0 THEN PU
     T(COLTAR,ROWTAR),TARGET%,XOR:COLTAR
     =COLTAR+5:PUT(COLTAR,ROWTAR),TARGET
     %,XOR:RETURN ELSE RETURN
1020 IF POINT(COLTAR-5,ROWTAR)=0 THEN PU
     T(COLTAR,ROWTAR),TARGET%,XOR:COLTAR
     =COLTAR-5:PUT(COLTAR,ROWTAR),TARGET
     %,XOR:RETURN
1030 IF POINT(COLTAR,ROWTAR+5)=0 THEN PU
     T(COLTAR,ROWTAR),TARGET%,XOR:ROWTAR
     =ROWTAR+5:PUT(COLTAR,ROWTAR),TARGET
     %,XOR:RETURN
1040 IF POINT(COLTAR+5,ROWTAR)=0 THEN PU
     T(COLTAR,ROWTAR),TARGET%,XOR:COLTAR
     =COLTAR+5:PUT(COLTAR,ROWTAR),TARGET
     %,XOR:RETURN
1050 IF POINT(COLTAR,ROWTAR-5)=0 THEN PU
     T(COLTAR,ROWTAR),TARGET%,XOR:ROWTAR
     =ROWTAR-5:PUT(COLTAR,ROWTAR),TARGET
     %,XOR:RETURN ELSE RETURN
1060 IF POINT(COLTAR,ROWTAR+5)=0 THEN PU
     T(COLTAR,ROWTAR),TARGET%,XOR:ROWTAR
     =ROWTAR+5:PUT(COLTAR,ROWTAR),TARGET
     %,XOR:RETURN
1070 IF POINT(COLTAR+5,ROWTAR)=0 THEN PU
     T(COLTAR,ROWTAR),TARGET%,XOR:COLTAR
     =COLTAR+5:PUT(COLTAR,ROWTAR),TARGET
     %,XOR:RETURN
1080 IF POINT(COLTAR,ROWTAR-5)=0 THEN PU
     T(COLTAR,ROWTAR),TARGET%,XOR:ROWTAR
     =ROWTAR-5:PUT(COLTAR,ROWTAR),TARGE
     T%,XOR:RETURN
1090 IF POINT(COLTAR-5,ROWTAR)=0 THEN PU
     T(COLTAR,ROWTAR),TARGET%,XOR:COLTAR
     =COLTAR-5:PUT(COLTAR,ROWTAR),TARGET
     %,XOR:RETURN ELSE RETURN
1100 IF POINT(COLTAR+5,ROWTAR)=0 THEN PU
     T(COLTAR,ROWTAR),TARGET%,XOR:COLTAR
     =COLTAR+5:PUT(COLTAR,ROWTAR),TARGET
     %,XOR:RETURN
1110 IF POINT(COLTAR-5,ROWTAR)=0 THEN PU
     T(COLTAR,ROWTAR),TARGET%,XOR:COLTAR
     =COLTAR-5:PUT(COLTAR,ROWTAR),TARGET
     %,XOR:RETURN
1120 IF POINT(COLTAR,ROWTAR+5)=0 THEN PU
     T(COLTAR,ROWTAR),TARGET%,XOR:ROWTAR
     =ROWTAR+5:PUT(COLTAR,ROWTAR),TARGET
     %,XOR:RETURN
1130 IF POINT(COLTAR,ROWTAR-5)=0 THEN PU
     T(COLTAR,ROWTAR),TARGET%,XOR:ROWTAR
     =ROWTAR-5:PUT(COLTAR,ROWTAR),TARGET
     %,XOR:RETURN
1140 '*** Intermission Time ***
1150 FOR X=1 TO 24:LOCATE X,1:PRINT SPAC
     E$(40);:NEXT X
1160 PLAY"o2t180mbl4gfefl1gl4gfefl1gl4g
     fefgfefgfe:POKE &H4E,1
1170 DEF SEG:POKE &H4E,
1180 LOCATE 1,11:PRINT "Congratulations!
     "
1190 LOCATE 4,1:PRINT "You have just com
     pleted maze"+STR$(MAZE)+". You are awarded an extra
     slither. PRINT "Please press the space
     bar to continue."
1200 DEF SEG:POKE &H4E,2
1210 LIVES=LIVES+1:LOCATE 25,26:PRINT LI
     VES;:
1220 SCOREC=SCOREC+500:MAZE=MAZE+1:CHANG
     EM=CHANGEM-10
1230 DEF SEG=0:POKE 1050,PEEK(1052)
1240 IF INKEY$<>"" THEN 1260 ELSE RETUR
     N
1250 '*** Introduction ***
1260 CLS:SCREEN 0
1270 COLOR 2,0,0:CLS
1280 LOCATE 8,13,0:PRINT CHR$(213)+STRIN
     G$(14,205)+CHR$(184)
1290 LOCATE 9,13:PRINT CHR$(179)+STRING$
     (14,32)+CHR$(179)
1300 LOCATE 10,13:PRINT CHR$(179)+" Wel
     come to"+CHR$(179)
1310 LOCATE 11,13:PRINT CHR$(179)+STRING
     $(14,32)+CHR$(179)
1320 LOCATE 12,13:PRINT CHR$(179)+STRING
     $(14,32)+CHR$(179)
1330 LOCATE 13,13:PRINT CHR$(179)+STRING
     $(14,32)+CHR$(179)
1340 LOCATE 14,13:PRINT CHR$(212)+STRING
     $(14,205)+CHR$(190)
1350 LOCATE 12,17:COLOR 20:PRINT "SLITHE
     R"
1360 COLOR 6:LOCATE 18,9
1410 COLOR 14:LOCATE 23,7:PRINT "Press E
     NTER to play the game"
1420 DEF SEG=0:POKE 1050,PEEK(1052)
1430 K$=INKEY$:IF K$="" THEN 1430 ELSE R
     ETURN
1440 '*** End of Game ***
1450 SCREEN 0:COLOR 3,0:CLS
1460 LOCATE 8,11:PRINT "Your score is:";
     SCORE
1470 LOCATE 12,4:PRINT "Do you want to p
     lay again (Y/N)?"
1480 DEF SEG=0:POKE 1050,PEEK(1052)
1490 K$=INKEY$:IF K$="N" OR K$="n" THEN
     END ELSE IF K$="Y" OR K$="y" THEN
     WICE=0:MAZE=0:SCREEN 1:COLOR 0,0:GO
     TO 430 ELSE 1490
```

HCM

SLITHER

TI-99/4A

```
 100 REM ********
 110 REM * SLITHER *
 120 REM ********
 130 REM BY AARON CHEW
 140 REM AND THE HCM STAFF
 150 REM HOME COMPUTER MAGAZINE
 160 REM VERSION 4.5.1
 170 REM TI BASIC
 180 REM
 190 RANDOMIZE
 200 OPTION BASE 1
 210 DIM SN(250,2),D(4,2)
 220 CALL CLEAR
 230 CALL SCREEN(4)
 240 FOR Z=1 TO 6
 250 READ A,A$
 260 CALL CHAR(A,A$)
 270 NEXT Z
 280 FOR Z=1 TO 4
 290 READ D(Z,1),D(Z,2)
 300 NEXT Z
 310 CALL COLOR(9,8,2)
 320 CALL COLOR(10,4,4)
 330 CALL COLOR(11,7,2)
 340 PRINT :::::::::::::::::::::::::TAB(
     10);"SLITHER"::::::
     PRESS ENTER TO CONTINUE"
 350 FOR Z=1 TO 8
 360 CALL COLOR(Z,16,2)
 370 NEXT Z
 380 CALL KEY(0,K,S)
 390 IF S=0 THEN 380
```

Continued

```
400 CALL LOAD(-31806,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0)
410 RESTORE 2290
420 LVS=1 :: Z=1 :: TO 4
430 FOR D=1 TO 4
440 READ DT
450 NEXT D
460 SPLS=3 :: SPLS=3 :: SPLS=3 :: SPLS=3
470 NOIXY=2 :: NOIXY=2 :: NOIXY=2 :: NOIXY=2
480 DSXY=8 :: DSXY=8 :: DSXY=8 :: DSXY=8
490 GOSUB 1880 :: GOSUB 1880 :: GOSUB 1880 :: GOSUB 1880
500 IFSUBV=1 :: IFSUBV=1 :: IFSUBV=1 :: IFSUBV=1
510 GOSUB 1400 :: GOSUB 1400 :: GOSUB 1400 :: GOSUB 1400
520 IFSUBV=2 :: IFSUBV=2 :: IFSUBV=2 :: IFSUBV=2
530 GOSUB 1400 :: GOSUB 1400 :: GOSUB 1400 :: GOSUB 1400
540 IFSUBV=3 :: IFSUBV=3 :: IFSUBV=3 :: IFSUBV=3
550 GOSUB 1400 :: GOSUB 1400 :: GOSUB 1400 :: GOSUB 1400
560 IFSUBV=4 :: IFSUBV=4 :: IFSUBV=4 :: IFSUBV=4
570 GOSUB 1400 :: GOSUB 1400 :: GOSUB 1400 :: GOSUB 1400
580 GOSUB 1400
590 GOSUB 1400
600 CALL HCHAR(2,5,112,3)
610 CALL HCHAR(SX,SY,95+DIR)
620 GOSUB 1420 :: GOSUB 1420 :: GOSUB 1420 :: GOSUB 1420
630 IFSUBT=(RND*25)<>9 :: THEN 650
640 GOSUB 1420
650 CALL KEY(0,K,S)
660 IF S=0 :: THEN 620
670 IF K=7 :: THEN 720
680 IF K=DIR :: THEN 760
690 IF K=DIR-5 :: THEN 760
700 IF K=7 :: THEN 620
710 IF K=DIR+5 :: THEN 760
720 IF K=DIR+1 :: THEN 620
730 IF K=DIR-1 :: THEN 760
740 IF K=4 :: THEN 760
750 IF K=4 :: THEN 760
760 CALL SOUND(10,440,0)
770 GOTO 620 :: GOTO 620 :: GOTO 620 :: GOTO 620
780 IFSUBC=1 :: THEN 810
790 SNC=SNC-1
800 SLA=SLA-1 :: SLA=SLA-1 :: SLA=SLA-1 :: SLA=SLA-1
810 IFSUBA=1 :: THEN 840
820 IFSUBA=2 :: THEN 840
830 IFSUBA=3 :: THEN 840
840 CALL HCHAR(SN(SLA,1),SN(SLA,2),32)
850 CALL HCHAR(SN(SP,1),SN(SP,2),112)
860 SP=SN(SP,1)
870 SN=SN(SP,2)
880 SP=SP+1
890 IFSUBP=25 :: THEN 910
900 SP=1
910 SP,1)=XPD(DIR,1)
920 SP,2)=YPD(DIR,2)
930 CALL HCHAR(SN(SP,1),SN(SP,2),CH)
940 IFSUBC=100 :: THEN 1550
950 IFSUBV=95 :: THEN 970
960 GOSUB 1000
970 CALL HCHAR(SN(SP,1),SN(SP,2),95+DIR)
980 CALL LOAD(50,-3,0)
990 RETURN
1000 CALL SOUND(10,220,10)
1010 CALL SOUND(30,330,8)
1020 CALL SOUND(60,440,6)
1030 CALL SOUND(100,660,4)
1040 CALL SOUND(200,880,0)
1050 SNC=SNC+10
1060 IFSUBCORE+20=INT(SCORE/500)*500 :: THEN 1140
1070 GOSUB 1420
1080 S$=STR$(SCORE)
1090 FOR L=1 TO LEN(S$)
1100 CALL HCHAR(10+Z,3,ASC(SEG$(S$,Z,1)))
1110 NEXT Z :: NEXT L
1120 GOTO 1420
1130 RETURN
1140 CALL SOUND(200,-3,0)
1150 FOR D=200 TO 2000 STEP 100
1160 CALL SOUND(-100,Z,0)
1170 NEXT D
1180 LVS=LVS+1 :: LVS=LVS+1 :: LVS=LVS+1 :: LVS=LVS+1
1190 IFSUBLV=5 :: THEN 1210
1200 LVS=4 :: NOSUBZ=1
1210 FOR D=1 TO LV
1220 GOSUB 1880 :: GOSUB 1880 :: GOSUB 1880 :: GOSUB 1880
1230 NEXT D
1240 GOSUB 1880
1250 NEXT Z :: NEXT L
1260 SLIDE=4
1270 DSPLAY#3:"SCORE:";SCORE
1280 DSPLAY#4:"LIVES:";LV
1290 SNC=0 :: SNC=0 :: SNC=0 :: SNC=0
1300 FOR D=1 TO 250
1310 NEXT D
1320 SNC=SNC-1 :: SNC=SNC-1
1330 NEXT D
1340 RESTORE 2290
1350 FOR D=1 TO 4
1360 READ DT
1370 NEXT D
1380 CALL HCHAR(2,5,112,3)
1390 CALL HCHAR(2,8,99)
1400 SUBEND
1410 SUBRN
1420 Z=1 :: TO 10
1430 RINDX=INT(RND*4)+1
1440 XYA=XYA(Z,1)
1450 XYA=XYA(Z,2)
1460 CALL HCHAR(EXA,EYA,CH)
1470 IFSUBTHEN 1500
1480 RETURN
1490 SUBRN
1500 CALL HCHAR(EX,EY,32)
1510 CALL HCHAR(EXA,EYA,95)
1520 EXA=EXA
1530 EYA=EYA
1540 RETURN
1550 FOR Z=10 TO 1 STEP -1
1560 CALL SOUND(SP,1),SN(SP,2),88)
1570 CALL SOUND(SP,1),SN(SP,2),43)
1580 CALL SOUND(SP,1),SN(SP,2),43)
1590 CALL SOUND(-100,-INT(RND*7)-1,INT(RND*7)-1,INT(RND*7)-1,INT(RND*7)-1)
1600 NEXT Z
1610 CALL LOAD
1620 SLA=SLA-1
1630 NOIXY=2 :: NOIXY=2 :: NOIXY=2 :: NOIXY=2
1640 SNC=0 :: SNC=0 :: SNC=0 :: SNC=0
1650 IFSUBTHEN 1220
1660 CALL SOUND(1,4)
1670 CALL SOUND(8,2,4)
1680 PRINT :: PRINT :: PRINT :: PRINT :: PRINT "SCORE:";SCORE :: PRINT :: PRINT :: PRINT :: PRINT "LIVES:";LV :: PRINT :: PRINT :: PRINT :: PRINT "PLAY AGAIN (Y/N)?"
1690 CALL KEY(0,K,S)
1700 IFSUBTHEN 1690
1710 IFSUBTHEN 2300
1720 IFSUBTHEN 1690
1730 CALL SOUND(1,4,2)
1740 CALL SOUND(8,16,2)
1750 SCORE=0 :: SCORE=0
1760 GOTO 490 :: GOTO 490
1770 SUBSTR$(SCORE)
1780 SYP=3 :: SYP=3
1790 GOSUB 1840 :: GOSUB 1840
1800 SXP=30 :: SXP=30
1810 SUBSTR$(NOS)
1820 GOSUB 1840 :: GOSUB 1840
1830 RETURN
1840 FOR P=1 TO LEN(S$)
1850 CALL HCHAR(PP+10,YP,ASC(SEG$(S$,P,1)))
1860 NEXT P
1870 RETURN
1880 FOR Z=7 TO 2 STEP -4
1890 CALL SOUND(Z,2,4)
1900 NEXT Z
1910 CALL HCHAR(24,1,104,64)
1920 CALL HCHAR(1,30,104,144)
1930 SXP=-LEN(STR$(SCORE))
1940 FOR L=1 TO LEN(S$)
1950 CALL HCHAR(2+Z,3,ASC(SEG$(S$,Z,1)))
1960 NEXT L :: NEXT Z
1970 SXP=-LEN(NOS)
1980 FOR L=1 TO LEN(S$)
1990 CALL HCHAR(Z+2,30,ASC(SEG$(S$,Z,1)))
2000 NEXT Z
2010 RETURN
2020 CALL HCHAR(4,19,104,9)
2030 CALL HCHAR(4,27,104,18)
2040 CALL HCHAR(21,6,104,22)
2050 CALL HCHAR(4,6,104,18)
2060 CALL HCHAR(4,6,104,9)
2070 RETURN
2080 CALL HCHAR(7,9,104,16)
2090 CALL HCHAR(7,24,104,12)
2100 CALL HCHAR(18,9,104,6)
2110 CALL HCHAR(7,9,104,12)
2120 RETURN
2130 CALL HCHAR(10,12,104,3)
2140 CALL HCHAR(10,19,104,3)
2150 CALL HCHAR(10,21,104,6)
2160 CALL HCHAR(15,12,104,10)
2170 CALL HCHAR(10,12,104,6)
2180 RETURN
2190 EXP=INT(RND*22)+2
2200 EYP=INT(RND*26)+4
2210 CALL HCHAR(EX,EY,CH)
2220 IFSUBTHEN 2200
2230 CALL HCHAR(EX,EY,95)
2240 RETURN
2250 DATA 96,183C7E5AFFFFFFFF7E,97,0E3F6FF0
2260 DATA FF0FE98C7E95FFFFFF5A7EFF0
2270 DATA FC12,3C0C,1,7,2,8
2280 DATA DATA
2290 DATA 1,5,2,6,2,
2300 END
```


HCM

DeBUGS on DisPlay

This issue of Home Computer Magazine debuts our updated "DeBugs on Display" format. Once all corrections and/or enhancements have been completed and tested in our programming laboratory, the new version of a program is compared to its previous version by our "cross-checking" computer. A listing of all the differences is produced, transmitted to the computerized typesetter, and formatted in the same fashion as our standard listings.

This new procedure for "DeBugs on Display" offers two advantages: (1) a standard presentation for updating your HCM programs that is clear and straightforward, and (2) inclusion of all published changes in "update files," which are placed ON DISK™ at the same time the corrections appear in print. This is of special significance to Apple, IBM, and TI (Extended BASIC programs only) ON DISK™ subscribers, because the correction file can be directly "merged" with the original file—automatically updating it! The procedures to accomplish this are included with the appropriate media.

We are currently working on a method of "update merging" for the Commodore 64, and hope to have it completed by next issue. (Yes, one method of merging program lines on the C-64 involves LISTing to the screen the lines of the correction file, and using the (RETURN) key to enter them into the original file—after it is loaded into memory. This is an error-prone task at best.)

If you are going to key-in the corrections from these pages directly into the original program, follow these steps:

1.) Load the original program into your computer's memory.

2.) Key-in the corrections line-by-line as directed in the "Program Typing Guide" at the beginning of the Listings section.

3.) Any lines in the listing of corrections that state "**** DELETED LINE," are to be deleted from the original program by entering the line number only and pressing either the (ENTER) or (RETURN) key (depending on your computer).

Each set of program corrections is prefaced by an identification bar that tells you the program name and the computer brand to which the correction applies. Make sure you are working with the right listing to ensure satisfactory results.

TAX DEDUCTION

HCM Vol. 4, No. 4 — TI-99/4A

```
150 REM VERSION 4.4.2
580 PRINT #2:"TOTALS REPORT": :"CATEGORY        TOTAL AMOUNT": : : FOR Z=1 TO 17 :: T(Z)=0 :: NEXTZ
```

ELEMENTARY ADD AND SUBTRACT

HCM Vol. 4, No. 3 — TI-99/4A

```
160 REM VERSION 4.3.2
945 U1=1
2345 QWL=ANSW
2780 IF U1=0 THEN 2570
2795 U1=0
2800 GOTO 2570
```

SNAP-CALC

HCM Vol. 4, No. 3 — TI-99/4A

```
150 REM VERSION 4.3.3
200 DEF L(M)=ASC(SEG$(E$(N),M,G)) :: FOR N=1 TO 500 :: NEXT N :: ON WARNING NEXT :: DV$="DSK1." :: FN$,K$="" :: PDV$="RS232.BA=9600.DA=8"
220 GOSUB 1210 :: IF ((P>47 AND P<58)OR P=45 OR P=46)AND CN>0 THEN GOSUB 1130 :: GOTO 220
405 IF U=204 AND J(N,T)=0 THEN J(N,T)=0 :: M=M+G :: GOTO 430
407 IF U=204 AND W=0 THEN J(N,T)=SGN(J(N,T))*99999.99 :: M=M+G :: GOTO 430
440 X$=X$&CHR$(X) :: NEXT N :: NEXT T :: IF F=0 THEN 475 ELSE DISPLAY AT(3,G):"        TOTAL COLUMN"
470 NEXT N
475 CALL SCREEN(6) :: H$=C$ :: GOSUB 1170 :: GOSUB 1260 :: O=0 :: RETURN
520 IF F$(G)="TOTAL" THEN IF F$(4)="OFF" THEN AV=G :: F=0 :: GOTO 510 ELSE F=VAL(F$(4)) :: LC=F-1 :: AV=0
525 IF F$(G)="TOTAL" THEN IF F>B THEN F=B :: LC=F-1 :: AV=0 :: PRINT "***OUT OF RANGE***" :: GOTO 510
527 IF F$(G)="TOTAL" THEN IF F<3 THEN F=3 :: LC=F-1 :: AV=0 :: PRINT "***OUT OF RANGE***" :: GOTO 510 ELSE GOTO 510
535 IF F$(G)="LAST" THEN PRINT "***OUT OF RANGE***" :: GOTO 510
750 I$="TOO MANY WORDS" :: GOSUB 1200 :: RETURN
930 CALL DELSPRITE(ALL) :: DISPLAY AT(1,1)ERASE ALL:TTL$:"ENTER YOUR CHOICE": :"1) DATA": :"2) LOGIC": :"3) ABORT FUNCTION"
940 ACCEPT AT(11,1)VALIDATE("123")SIZE(1):LM :: CALL CLEAR :: ON LM GOTO 1030,950,945
945 O=0 :: ON ERROR 1550 :: RETURN
960 IF DV$<>"CS1" THEN DISPLAY AT(13,1):"ENTER FILE NAME:":K$ :: ACCEPT AT(14,1)SIZE(-28):TEMP$
970 TEMP$=SEG$(TEMP$,1,10) :: IF DV$<>"CS1" THEN DVS$=SEG$(DV$,1,4)&"."&TEMP$ ELSE DVS$=DV$
980 ON ERROR 1640 :: IF P=3 THEN OPEN #G:DVS$,INPUT ,INTERNAL,FIXED 192 ELSE OPEN #G:DVS$,OUTPUT,INTERNAL,FIXED 192
1000 INPUT #G:TEMP$,S,F,LC,A,B :: FOR N=1 TO A :: INPUT #G:D$(N),E$(N) :: NEXT N :: FOR N=1 TO 50 :: INPUT #G:K(N) :: NEXT N :: O=0
1005 CLOSE #G :: ON ERROR 1550 :: K$=TEMP$ :: RETURN
1015 CLOSE #G :: ON ERROR 1550 :: K$=TEMP$ :: RETURN
1040 DISPLAY AT(12,1):"FILE NAME:":FN$ :: ACCEPT AT(13,1)SIZE(-28):FT$ :: FX$=SEG$(FT$&"--------",1,9)&"*" :: DVS$=SEG$(DV$,1,4)&"."&FX$
1050 ON ERROR 1640 :: IF P=3 THEN 1080
1070 PRINT #G:J(N,8);J(N,9);J(N,10);J(N,11);J(N,12);J(N,13) :: NEXT N :: CLOSE #G :: O=0 :: ON ERROR 1550 :: FN$=FT$ :: RETURN
1090 INPUT #G:J(N,8),J(N,9),J(N,10),J(N,11),J(N,12),J(N,13) :: NEXT N :: CLOSE #G :: O=0 :: ON ERROR 1550 :: FN$=FT$ :: RETURN
1130 IF P=45 THEN J(RN,CN)=-J(RN,CN) :: DISPLAY AT(R*2+3,(Q-1)*8+6)SIZE(7):USING 1500:J(RN,CN) :: RETURN
1135 IF AC<>G THEN AC=G :: M$="0" ELSE IF LEN(M$)=8 OR VAL(M$&CHR$(P))>9999.99 THEN CALL SOUND(100,220,0) :: RETURN
1450 NEXT TP :: PRINT #5: :TAB(5);"|";RPT$("-",125):TAB(5);"|": :FOR N=G TO MIN(NR,A) :: IF LEN(D$(N))>0 THEN IF ASC(D$(N))=42 THEN 1490
1460 PRINT #5:N;TAB(5);"|"; :: FOR M=0 TO 13 :: IF PG=1 AND M=0 THEN PRINT #5,USING 1520:D$(N); :: GOTO 1470
1465 PRINT #5,USING 1510:J(N,(PG-1)*14+M);
1630 REM DISK ERROR HANDLER
1640 PRINT "FILE CAN'T BE OPENED" :: CALL SOUND(200,110,0) :: FOR XZ=1 TO 500 :: NEXT XZ :: RETURN 920
```

SNAP-CALC HCM Vol. 4, No. 3

APPLE // Family

SNAP-CALC HCM Vol. 4, No. 3

IBM PC & IBM PCjr

DEBUGS CONCLUDE ON NEXT PAGE

SNAP-CALC

HCM Vol. 4, No. 3

```
 160 REM  VERSION 4.3.5
 260 IF(K=134AND SC<>0)ORK=45ORK=46OR(K>4
     8AND K<58AND SC<>0)THENGOSUB2770
 590 FOR M=1 TO LEN(E$(N))-1:U=ASC(MID$(
     E$(N),M,1))
 690 IF U<>204 THEN 710
 692 IF VAL(D$(I,N))=0 THEN GOTO760
 694 IF W=0 THEN D$(I,N)=STR$(99999.99*S
     GN(VAL(D$(I,N)))):GOTO760
 720 IF" 99999.99"VAL(D$(I,N))>99999.99THEND$(I,N)
     =" 99999.99"VAL(D$(I,N))<-99999.99THEND$(I,N
     )="-99999.99"
 730 IF" 99999.99"VAL(D$(I,N))<-99999.99THEND$(I,N
     )="-99999.99"
 740 ********DELETED LINE
 750 ********DELETED LINE
 780 IFTC=N=1 THEN8 RL:D$(TC,N)=""
 782 FOR********DELETED LINE
 800 ********DELETED LINE
 810 ********DELETED LINE
 820 IF V"99999.99"VAL(D$(TC,N))>99999.99THEND$(TC,
     N)="-99999.99"
 830 IFV=VAL(D$(TC,N))<-99999.99THEND$(TC,
     N)="-99999.99"
 970 GOFSUB43600:IF ERR=1 THEN1220
 975 IFVAL(F$(4))<30RVAL(F$(4))>CLTHEN12
     30
1020 MAX=TC-1:IF TC=0 THEN MAX=CL
1030 IFVAL(F$(4))<1ORVAL(F$(4))>MAXTHEN1
     230
1040 LC=VAL(F$(4)):GOTO920
1740 PRINT"1 SAVE DATA":PRINT"2 SAVE
     LOGIC":PRINT"3 ABORT SAVE":PRINT
1780 ********LINE
1790 GOSUB32900:RETURN
1835 OPEN15,8,15:PRINT#15,"S0:"+DF$:CLOS
     E15
1840 OPEN1,8,8,DF$+",D,"+",S,W":GOSUB1860
     :CLOSE1:RETURN
1950 GOSUB25800:PRINT:PRINT
2125 FORJ=1TOLEN(T$)
2126 IFMID$(T$,J,1)=CHR$(13)THENT$=LEFT$
     (T$,J-1)+CHR$(206)+RIGHT$(T$,LEN(T$
     )-J)
2127 NEXTJ
2170 PRINT"3 ABORT LOAD":PRINT
2210 ********DELETED LINE
2220 GOSUB32900:RETURN
2555 T$=ES$(I):FORJ=1TOLEN(T$)
2556 IFMID$(T$,J,1)=CHR$(13)+RIGHT$(T$,LEN(T$
     )-J)
2557 NEXT:ES$(I)=T$
2775 N=0:I"K>45ANDK<58ANDK<>47THENDS$(SC,
     R))=STR$(-VAL(DS$(SC,SR))):GOSUB4330:IFI=0THENRETURN
2780 IFK=45THENDS$(SC,SR)=STR$(-VAL(DS$(SC
     ,SR)))):GOSUB4330:IFI=0THENRETURN
2786 IFK=45THENGOSUB32700:GOTO2780
2787 IFK=46ANDDP>0THENGOSUB32700:GOTO2780
2788 IFK=46THEN"."DP=1:GOSUB32700:GOTO2780
2790 IFK<>13ANDK<>27:GOSUB32700:I=0:GOSUB43
     30:GOTO2790
2795 IFK=13ANDK<>145ANDK<>157THENRETURN
2800 IFK=13ORK=17ORK=29ORK=145ORK=157THE
     NGOSUB35200:RETURN
2810 IFK<>45ANDK<>46ANDK<>48ORK>57)THE
     N:GOSUB32700:GOTO2780
2812 IFDP=0ANDVAL(DS$(SC,SR))+CHR$(K))>999
     999.99THENGOSUB32700:GOTO2780
2814 IFDP=0ANDVAL(DS$(SC,SR))+CHR$(K))<-99
     999.99THENGOSUB32700:GOTO2780
2818 IFDP=1THENDP=2:IFNO>2THENGOSUB32
     700:GOTO2780
2820 ********DELETED LINE
2824 IFI=0THENI=1:DS$(SC,SR)=CHR$(K):GOSU
     B43330:GOSUB32700:GOTO2780
2830 DS$(SC,SR)=DS$(SC,SR)+CHR$(K):GOSUB43
     30:GOTO2780
2840 ********DELETED LINE
2850 ********DELETED LINE
2860 ********DELETED LINE
2870 ********DELETED LINE
2880 ********DELETED LINE
2890 ********DELETED LINE
2900 ********DELETED LINE
2910 ********DELETED LINE
2920 ********DELETED LINE
2930 ********DELETED LINE
2940 ********DELETED LINE
2950 ********DELETED LINE
2960 ********DELETED LINE
2970 ********DELETED LINE
2980 ********DELETED LINE
2990 ********DELETED LINE
3000 ********DELETED LINE
3010 ********DELETED LINE
3020 ********DELETED LINE
3030 ********DELETED LINE
3040 ********DELETED LINE
3050 ********DELETED LINE
3060 ********DELETED LINE
3070 ********DELETED LINE
3080 ********DELETED LINE
3090 ********DELETED LINE
3100 ********DELETED LINE
3110 ********DELETED LINE
3120 ********DELETED LINE
3130 ********DELETED LINE
3200 IFASC(QR$)=20ANDLS<>""THENPRINT"
     CRSR LEFT;:LS=MID$(LS,2):GOTO3200
3300 PRINT:GOSUB31400:PRINT"ROW";:P
     RINT"COL";:INPUT"ITEM";I:IFI=0THEN
     3300
3330 IFI<1ORI>TO 10:X=0:Y=(I+1)*2:GOSUB46
     010:GOTO3300
3395 SC:TR=SR
3420 Y=0:J=2+2:POKE781,Y:POKE782,
     SR+J-1:SYS65520:IFVAL(DS$(SC,SR))=
     0.00"":GOTO34300
3427 GOSUB43400:SC=ZC:SR=TR:RETURN
3430 ********DELETED LINE
3450 ********DELETED LINE
3460 ********DELETED LINE
3470 ********DELETED LINE
3480 ********DELETED LINE
3490 ********DELETED LINE
3500 ********DELETED LINE
3510 ********DELETED LINE
3530 N=0:IFVD=0:IFK<>17ANDK<>
     29ANDK<>32ANDK<>13ANDK<>34THENVD=1:RETURN
3590 IFSC<>0THENVD=1:RETURN
3595 IFSC=0ANDLS=""THENVD=1:RETURN
3800 IFFLEN(LS)=1THENGOSUB43400:IFVAL(DS$(SC,SR))=0"":RETURN
3900 IFRIGHT$(LS,1)=CHR$(34)THENLS=LEFT$(LS,LEN(LS)-1):GOTO3900
4230 PRINT"1 DATA":PRINT"2 ERASE DATA":PRINT
4235 PRINT"3 ERASE BOTH DATA AND LOGIC
4270 PRINTQ1$:ONVAL(Q1$)GOSUB42900,43000
4290 FORI=1TOTO:IFSC<>0THENRL:DS$(I,J)=""
4300 R=NEXTJ:NEXTI:RETURN
4330 IFSC=0THENRL:ES$(I)="":RETURN
4334 ((SX+(PEEK(V+16)*255)-100)/88+1)*
4335 (SY-75)/16+2)*2+STR$(INT(ABS(VA
4340 DS$(SC,SR))))*SGN(VAL(DS$(SC,SR))))
4341 GN(VAL(DS$(SC,SR))))=-1ANDDP$=""
4342 INT((ABS(VAL(DS$(SC,SR))))*100+.1)
4344 (VAL(STR$(VV)),LEN(STR$(VV)))-1
4346 LEN(P1$)=1THENP1$="0"+P1$
4350 POKE781,Y:POKE782,X:POKE783,0:SYS65
     520:PRINT:RETURN
4410 V=53248
4440 POKE 204,11:POKE V,SX:POKE V+1,SY
```

MARKET MADNESS

HCM Vol. 4, No. 4

IBM PC & IBM PCjr

```
 160 'VERSION 4.4
 640 IF SP$=NOS>PC(TN,1) THEN LOCATE 24,1
     :PRINT DOES'T HAVE ENOUGH
     MONEY";:GOTO620"SIZE OF LOAN";A$
1460 LOCATE 13,1:IF TATE=15,1:PRINT"
     :GOSUB16980:ELSE
     :ELSE WEEKLY PAYMENT IS:";R
     :PRINT"PRESS [ENTER] TO CONTINUE
     :PRINT:GOSUB 16980:RETURN
```

COLLECT ALL BACK ISSUES

HOME COMPUTER™ magazine

Please Print

Name _____

Address _____

City _____ State _____ Zip _____

☐Check or Money Order Enclosed **Total** _____
MUST BE IN U.S. FUNDS DRAWN ON A U.S. BANK

Bill my ☐VISA ☐MasterCard Date Expires _____

Account No. | | | | | | | | | | | | | | | | |

Tel. No. _____ Signature _____

Enclose payment or credit card information & mail with completed form to:

Home Computer Magazine
P.O. Box 5537 ● Eugene, OR 97405

Or use our TOLL-FREE Order Line for VISA/MasterCard orders only:
1-800-828-2212
In Oregon, Alaska, Hawaii Tel. (503) 485-8796

ITEMS	PRICE
Home Computer Magazine Back Issues (Circle Issues Desired) Vol. 4 No. 1 Vol. 4 No. 2 Vol. 4 No. 3 Vol. 4 No. 4 Vol. 4 No. 5	$3.95 each — U.S. $4.50 each — Canada $5.50 each — Foreign Surface $7.50 each — Foreign Air
ON DISK & ON TAPE Back Issues (Circle Issues Desired) Vol. 4 No. 1 Vol. 4 No. 2 Vol. 4 No. 3 Vol. 4 No. 4 Vol. 4 No. 5	$5.95 each — U.S. $7.95 each — Canada $9.95 each — Foreign Air
SAVE EVEN MORE — Order Combined Sets (Circle Magazine & Media Sets Desired) Vol. 4 No. 1 Vol. 4 No. 2 Vol. 4 No. 3 Vol. 4 No. 4 Vol. 4 No. 5	$7.90 each set — U.S. $10.90 each set — Canada $10.90 each set — Foreign Surface

Indicate your choice of media: ☐ON TAPE™ ☐ON DISK™
Indicate which computer media is for: (check one)
☐Apple ☐C-64 ☐IBM PC ☐IBM PCjr ☐TI-99/4A
Defective media gladly exchanged. NO REFUNDS on media.

For more information see inside front cover.

Give A GIFT SUBSCRIPTION To HOME COMPUTER™ magazine

Please Print

PLEASE ENTER A 1-YEAR GIFT-SUBSCRIPTION FOR:

Name _____

Address _____

City _____ State _____ Zip _____

FROM: (Gift-giver must complete this section)

Name _____

Address _____

City _____ State _____ Zip _____

☐ **Check if you would like a gift card sent in your name.**

☐ **Enclosed is $25* for each 1-year gift-subscription**
Add $11 per year for Canada; Foreign add $21 per year.
Free software offer available in U.S. & Canada only.

* Offer & Prices Subject To Change Without Notice.

☐Check or Money Order Enclosed **Total** _____
MUST BE IN U.S. FUNDS DRAWN ON A U.S. BANK

Bill my ☐VISA ☐MasterCard Date Expires _____

Account No. | | | | | | | | | | | | | | | | |

Tel. No. _____ Signature _____

The Perfect Gift For Any Occasion!

PLUS A BONUS!

We will send the gift-giver or the gift-recipient a CERTIFICATE valid for 2 months worth of software for each 1 year subscription.

Please indicate who receives this certificate:
☐ Gift-Giver ☐ Gift-Recipient

Enclose payment or credit card information & mail with completed form to:

Home Computer Magazine
P.O. Box 5537 ● Eugene, OR 97405

Or use our TOLL-FREE Order Line for VISA/MasterCard orders only:
1-800-828-2212
In Oregon, Alaska, Hawaii Tel. (503) 485-8796
Allow 6-8 weeks for your first issue.

For more information see page 8.

Save $ $ $ On BACK ISSUES of 99'er HOME COMPUTER™ magazine

SPECIAL CLOSE-OUT PRICES FOR MAGAZINES, DISKS & TAPES NOW IN EFFECT FOR TI-99/4A USERS!

Please Print

Name _____

Address _____

City _____ State _____ Zip _____

☐Check or Money Order Enclosed **Total** _____
MUST BE IN U.S. FUNDS DRAWN ON A U.S. BANK

Bill my ☐VISA ☐MasterCard Date Expires _____

Account No. | | | | | | | | | | | | | | | | |

Tel. No. _____ Signature _____

Enclose payment or credit card information & mail with completed form to:

Emerald Valley Publishing Co.
P.O. Box 5537 ● Eugene, OR 97405

Or use our TOLL-FREE Order Line for VISA/MasterCard orders only:
1-800-828-2212
In Oregon, Alaska, Hawaii Tel. (503) 485-8796

99'er Home Computer Magazine, Disk, & Tape Back Issues
Exclusively For The TI-99/4A Home Computer

ISSUE	Magazine Only	Mag. & Media Set	Media Only (Disk or Tape)	ISSUE	Magazine Only	Mag. & Media Set	Media Only (Disk or Tape)	ISSUE	Magazine Only	Mag. & Media Set	Media Only (Disk or Tape)
Vol. 1 No. 6				Mar. '83				Aug. '83			
Nov. '82				Apr. '83				Sept. '83			
Dec. '82				May '83				Oct. '83			
Jan. '83				June '83				Nov. '83			
Feb. '83				July '83							

☛ INDICATE CHOICE OF MEDIA: ☐ DISK ☐ TAPE

Place an "X" in the corresponding box for each item you wish to order.

QTY	MAGAZINE PRICES U.S.	Canada	Foreign	MAG/MEDIA SET PRICES U.S.	Canada	Foreign	MEDIA PRICES U.S.	Canada	Foreign
1	$3.95	$4.50	$5.50	$7.50	$9.50	$10.50	$3.95	$5.95	$6.95
3	5.95	7.50	8.95	14.95	17.95	18.95	10.95	13.95	14.95
6	10.95	12.50	13.95	28.50	33.50	34.50	20.95	26.95	27.95
12	21.90	24.00	25.95	49.95	55.95	56.95	39.95	44.95	45.95

SUB TOTAL $ _____

Defective media gladly exchanged. NO REFUNDS on media.

For more information see inside back cover.

GUIDE TO HOME COMPUTER™ magazine READER SERVICES

HOME COMPUTER™ magazine

See Rear Bind-In Card

Subscriptions

See Page 8

Gift-Subscriptions

See Inside Front Cover

Back Issues

ON TAPE™  ON DISK™ 

See Rear Bind-In Card

This Issue's Software

See Rear Bind-In Card

Program Subscriptions

See Inside Front Cover

Back Issues

99'er HOME COMPUTER magazine™

See Inside Back Cover

Back Issues

See Page 85

Blank-Media Service

LIMITED SUPPLY

A Back-Issue/Software Bonanza Of 99'er HOME COMPUTER magazine™ At Unbeatable Prices

The original 99'er Magazine and 99'er Home Computer Magazine were the forerunners of the present-day Home Computer Magazine. Each of these magazine back issues—exclusively covering the Texas Instruments TI-99/4A—is now available with your choice of either a floppy disk or a cassette tape that contains all the programs in that issue.

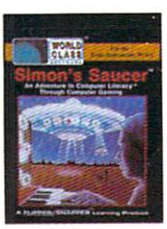
14 MAGAZINE & MEDIA SETS

AS LOW AS $1.83 EACH MAGAZINE!
See Order Card at Center of Magazine
ABOUT $5 PER SET!

MAGAZINES OR MEDIA MAY BE ORDERED SEPARATELY
To Order, Use Bind-in Card At Center Of Magazine.

SPECIAL MIX OR MATCH BONUS
—WHILE SUPPLIES LAST—

You will receive a FREE Simon's Saucer™ package and a FREE TI-FEST™ commemorative poster when your order includes at least 12 items from this page in any combination of magazines or media.



- A quality, ready-to-run game on cassette tape.
- A durable and attractive ring-binder collector's case for your software library.
- A complete, easy-to-use programming lesson on a deck of colorful flip cards.

SAVE UP TO $44 AND RECEIVE FREE GIFTS WORTH OVER $15!

Issues No. 1-5 OUT OF PRINT
Contents available in book form as "Best of 99'er"—Vol. 1.
See Home Computer Digest or inquire

ISSUE #6 (Partial Contents)
• How To Produce Sound Effects • Debugging a Game Program • How to Start a User's Group • Verbosk: A Speech Vocabulary Expansion Aid • Color Mapping • Dynamic Manipulation of Screen Character Graphics • The Beginner's Guide to Cassette Operation With the Home Computer • Pre-School Block Letters and Data Compaction • Picking the Ponies in TI BASIC • Battle Star Space Game • 3-D Animation on the Home Computer • Programming Tips • Who is LOGO for? • Tower of Hanoi in TI LOGO • A Review of the TI Lesson Development Software • An Interview with a Game Designer • Learning Assembly Language with a Magic Crayon • and much, much more.

NOVEMBER 1982 (Partial Contents)
• Chatting with Your Micro: Languages for the Home Computer • A Review of the Smith Corona TP-1 Daisy Wheel Printer • The Micro Jaws Arcade Game • A Knight's Tour in TI BASIC • LOGO Has Style • ASPIC: A Language for Children • A p-System Beginners' Tutorial • An Interview with a p-System Pioneer • A Mini-Memory Screen Dump to the Home Computer Printer • Up Scope!—An exciting Undersea Combat Game • Strategy for Munch Man • A Brief Encounter with a TI Hand-Held Computer • 99'er Shopping Bus • A Pocket Battleship • Sub-Programs in Extended BASIC • Arcade & Adventure Game Reviews • and much, much more.

DECEMBER 1982 (Partial Contents)
• Tex-Scribe: A Text Editor for the Home Computer • A Christmas Computer Carol • Managing a Mailing List the Futura Way • Parsec: The Arcade Game • Plotting With the Home Computer—Pixel by Pixel • Preventing the Situation—Oh No! Memory Full • A Colorful Tour of TI-Fest: The Home Computer Show • Santa's Workshop: The Making of a Home Computer • The Turtle Arcade: Movies & Video Games in LOGO • Controlling a BASIC Termite • The 99'er Gold Rush—An Arcade/Adventure in the Home • 99'er Digest of News & Happenings in the TI World • Plus Games, Reviews, and much, much more.

JANUARY 1983 (Partial Contents)
• Computer Assisted Instruction for the Handicapped • p-System Basics • Debugging in LOGO • The Dow-4 Gazelle Flight Simulator • Note Whiz and Pitch Master Musical Game Reviews • Learning With the PLATO Computer Library • Strategies for Adventure Gaming • Death Drones • Using the Line-By-Line Assembler • Close Encounters of the Simon Kind • Electrical Engineering Education Program • Interview With an Arcade Game Designer • TI Invaders • Programming With Pascal • Cyber-Dice • News and Happenings in the Home Computer World • Arcade Game Reviews • The Thief Adventure Game • Programming Tips • and much, much more.

FEBRUARY 1983 (Partial Contents)
• Texas Instruments at the Winter Consumer Electronics Show • Home Computer Printers on Review • How to Create Math Daisies in LOGO • Vectors in LOGO • ASPIC: A Language for Teachers • The Joys of Adventuring—Part 2 • Programming Pointers with Chuck-A-Luck—Part 4 • Interview With the Voice of Parsec • Why You Need a Printer for Your Home Computer • Lifeline to Titan Space Game • Night Blockade Battleship Game • Tower of Hanoi Pocket Program • Computer Gaming Software Reviews • News of Late Developments in the World of Home Computers • and much, much, more.

MARCH 1983 (Partial Contents)
• An Introduction to the TI 99/2 Basic Computer • The Hex-bus and the 4/A Connection • Making Your Own Say and Spell Game • Disabled Children Learn and Grow • Super Cataloger—A Review of a Disk Library Utility Program • TI's New CC-40 Compact Computer • Robots and Their Social Impact • Twenty Questions With Robot Redford • The Gravity of LOGO • Joystick Jockey—An Overview of Remote Controllers • Parsec Strategy • Converting Extended BASIC to Assembly Language • Matrix Muncher • Mini Memory Disassembler Utility • Pulling the Shade on Sprites • Letters on LOGO • Tiny Tutorials • Games programs, reviews, and much, much more.

APRIL 1983 (Partial Contents)
• Computer Assisted Savings Planning to Build Your Nest Egg • Tex-Cipher Writes and Decodes Secret Messages • Crossbytes—Computer Vocabulary Crossword Puzzle • Cutting Corners On Your Food Budget Using Coupons • Introducing Financial Planning with Multiplan • The Design Philosophy of the Compact Computer • LOGO Takes On the Popular Fifteen Puzzle • Super Language—Programming Sprites in Mini Memory • Colorful Word World—Reading Readiness for Pre-schoolers • Gameware Buffet's A Maze-ing Boa Alley Game • Giant and Dwarfs Entrapment Game • Game Reviews • Programming Tips • Money Saving Hints • and much, much more.

MAY 1983 (Partial Contents)
• A Consumer's Guide to Word Processing • Word Processing Market Basket • A Generalized Filing Program for VIPs • The Multiplan Medium Balances Your Checkbook and Budget • Activity Accountant Helps School Secretaries with Extracurricular Activities • Maximizing Your Mini Memory's 4K of RAM • Exploring Enhanced BASIC on the Compact Computer • The LOGO Tortoise Debates the BASIC Hare • A Pocket Program to Organize Data with Linked Lists • Mentally Handicapped Learners Team Up with the TI-99/4A • The Wonders of Diskette Storage • Beeline—a Multi-Screen Strategy Game • Lost Ruins—an Archeological Dig Game • 3-D Illusions with Sprites in Depth • Game reviews, Group Grapevine, and much, much more.

JUNE 1983 (Partial Contents)
• Children and Computers Make the 99/4H Connection • Tune Your Guitar with Our TI Tuning Fork • Talk to Your Computer—Voice Technology is Here • Gameware Buffet's Eat or Be Eaten Aardvark Game • Protect Your Station in the Space Zapper Game • What Multiplan Can and Can't Do • Understanding Inputs and Outputs in Drive For Diskettes—Part 2 • Calculate Loan Schedules on the CC-40 • Go on a LOGO Vacation • Letters on LOGO • A Review of Upper Room Software's Programs for Special Learners • Construct an RS232 Joystick Interface • Group Grapevine • Shopping Bus • A Natural Language Interface for the Professional • Game Reviews • and much, much more.

JULY 1983 (Partial Contents)
• The Evolution of Home Computer Graphics Comes Alive in Graphics Grows Up • Five Data Organizers in Never Out of Sorts • TI & 99'er at the Consumer Electronics Show • WarGames: The Movie and the Book • Editing with Multiplan • The LOGO Logician Presents To Model Is to Learn • LOGO Mosaic Designs Fill the Screen • Your Speech Synthesizer as a Spelling and Foreign Language Teacher • Software for Your Low-cost Printer Port • Gameware Buffet's Treasure Island and the Colorful Switch-A-Row • A Book Review of Learn BASIC for CC-40 Users • 3-D Animation with the TMS 9918A Video Chip • Games Reviews • Group Grapevine • and much, much more.

AUGUST, 1983 (Partial Contents)
• The Home Computer Goes To Work • Bit One, Part Two at the Fashion Factory • Better Business Bar Graphs in Graphic Persuasion • An Ensemble of Assemblers • Cashflow Helps Money Management • Keystrokes for Thrifty Folks—a Review of Typewriter • Game Reviews of Cavern Quest and Starprobe '99 • Counting Fun for Preschoolers • Peripheral Vision 99 • Mean Machines and Small Populations • Multiplan Medium Groups Cells into Rectangles • Turtle Text, a LOGO Word Processor • Group Grapevine • Hello, Little Brother—CC-40 Speaks to 99/4A • Plato's Progress Looks at Student Assessment • Pocket Sunrise • Public Investigator • Gameware Buffet's Jungle Jim, Success Formula, and much, much more!

SEPTEMBER, 1983 (Partial Contents)
•Adding on to Your Home Computer • 99'er Directory of Commercially Available Software, Accessories and Peripherals • Peripheral Vision 99 Hardware Reviews • Pocket Sunrise, Part Two in Extended BASIC • Byte Lightning—a Mini Memory Plays a Game • TI-WRITER Tutorial • LOGO's Functions, Sets and Turtles • The CC-40 and 99/4A Take the Data, and RUN • PLATO's Progress Looks at Geometry Courseware and the Shape of Things to Come • Gameware Buffet's Challenge of Camelot and The Fly • Game Reviews of Nit Wit and Crime and Punishment • Group Grapevine • 99'er Hall of Fame • 99'er Digest Update on New Products, and much, much more!

OCTOBER, 1983 (Partial Contents)
• Adventures and Fantasy with Your Home Computer • Do-It Yourself Adventure • Picoprocessor Emulator, a 4-Bit Microprocessor • Have No Fear: Assembly Language Won't Byte, Part 1 • Make Your Mark • Les Izmore and Debug Computing Cartoon • Once Upon a Tortoise Shell—A LOGO Adventure • Turtle Strut • Number Nibbler for Children • Lots of Plots on Your CC-40 and HX1000 Four Color Printer/Plotter • Multiplan Medium—Bartender • A Grisly Adventure Bear Hunt • Escape From Wizard's Keep in Extended BASIC • Game Reviews • Shopping Bus • and much, much more.

NOVEMBER, 1983 (Partial Contents)
• Education with your Home Computer • Five Creative Learning Activities for Children • Let's Build America • Have No Fear: Assembly Language Won't Byte, Part 2 • Squeezing the Most Out of TI BASIC • TI-WRITER Tutorial • LOGO Lexicon • Interview with Dale Osborn • TI-WRITER: At Home in the Office • PLATO's Progress • Les Izmore and Debug • The Multiplan Medium • Computer Arrested Instruction—99'er Interviews the Kids • Gameware Buffet: Taco Man and Robo Chase • Game Reviews of Jail Break and Arithmetex • Hall of Fame • 99'er Digest • and much, much more.

Hurry—Supplies Are Limited. Order Yours Today!

ALL PROGRAMS IN THIS MAGAZINE

ONLY $3.95* DELIVERED RIGHT TO YOUR DOOR!

The same high-quality Apple, Commodore, IBM, and Texas Instruments programs with type-in-and-RUN listings in this issue are now available on ON DISK™ or ON TAPE™ to newsstand purchasers or subscribers of this magazine.

For only $3.95* (barely covering the cost of a blank floppy disk or cassette tape), you receive all the programs for your particular brand of computer—truly a "Software Giveaway!"

To Order, Use The Bind-In Card Inside Rear Cover.

* Current Issue Price Only — See Center Bind-In Card For Back Issue Prices.