SIMI VALLEY, CA 93065
356 LAGUNA TERR
BYRON JOHNSON MY81
93065JOHNS809 D1 2251

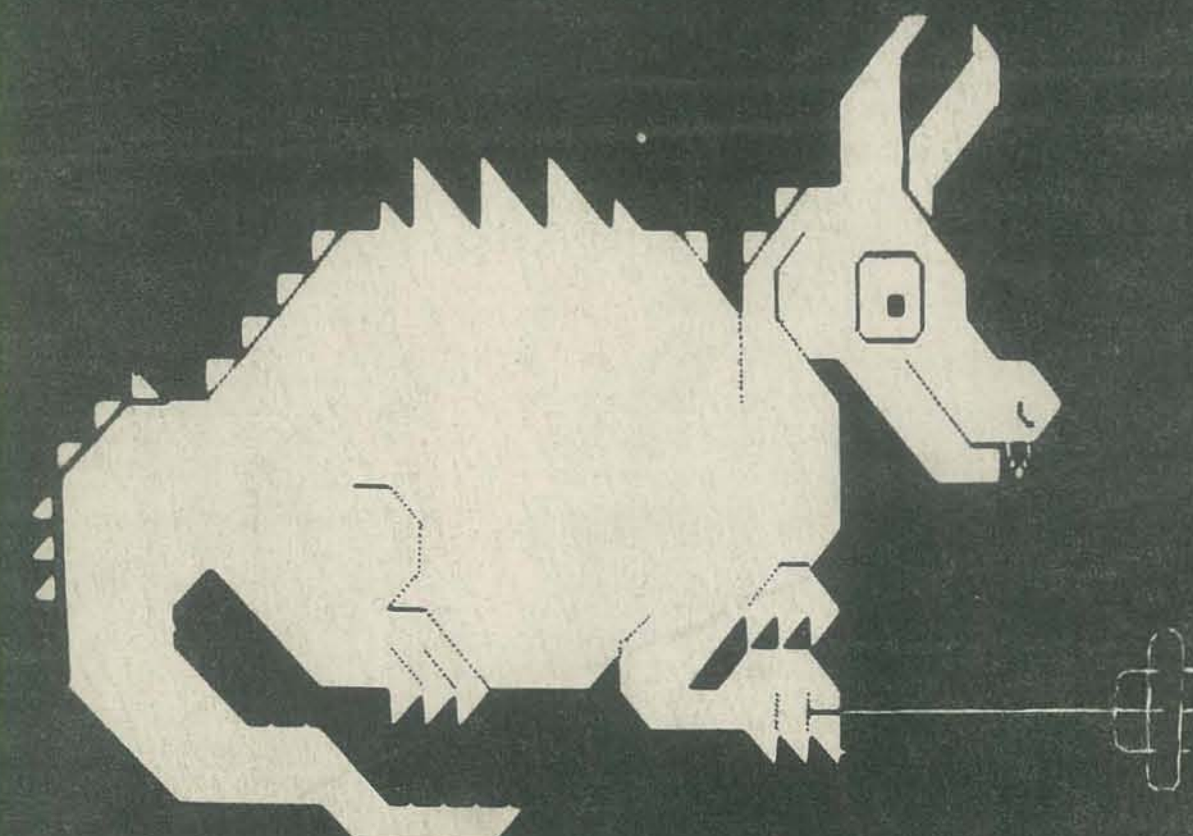
Second Class Postage Paid at Menlo Park, CA
Address Correction Requested

FORMERLY PEOPLE'S COMPUTERS In Canada $2.50 $2.00

Recreational COMPUTING

VOL 8 NO 3 ISSUE 43 NOV-DEC 1979

DRAGON CONTEST RESULTS!



SHOGI: GAMES FOR YOU TO PROGRAM
ATARI SOUNDS • CRYPTARITHMS
TEXAS INSTRUMENTS GRAPHICS

SUBMITTING ITEMS FOR PUBLICATION

LABEL everything with your name, address and the *date*; tapes should also include the program name, language and system. TYPE text if at all possible, double-spaced, on 8½ x 11 inch white paper. DRAWINGS should be as clear and neat as possible in black ink on white paper.

LISTINGS are hard to reproduce clearly, so please note:

- Use a new ribbon on plain white paper when making a listing; we prefer roll paper or fan-fold paper.
- Send copies of one or more RUNs of your program, to verify that it runs and to provide a sense of how things work—and to motivate more of us to read the code. RUNs should illustrate the main purpose and operation of your program as clearly as possible. Bells, whistles and special features should just be described in the documentation unless they're particularly relevant.
- Make sure your code is well documented—use a separate sheet of paper. Refer to portions of code by line number or label or address, please, not by page number. When writing documentation, keep in mind that readers will include beginners and people who may be relatively inexperienced with the language you're using. Helpful documentation/annotation can make your code useful to more people. Documentation should discuss just which cases are covered and which aren't.
- If you send us a program to publish, we reserve the right to annotate it (don't worry, we won't publish it if we don't like it).
- Last but not least, please try to limit the width of your listings: 50-60 characters is ideal. Narrow widths mean less reduction, better readability and better use of space.

LETTERS are always welcome; we assume it's OK to publish them unless you ask us not to. Upon request we will withhold your name from a published letter, but we will not publish correspondence sent to us anonymously. We reserve the right to edit letters for purposes of clarity and brevity.

ADVERTISING

SUBSCRIPTION INFORMATION

U. S. RATES

- ☐ $10/1 yr. (6 issues)
- ☐ Retaining subscription @ $25 ($15 tax deductible)
- ☐ Sustaining subscription @ $100+ ($90+ tax deductible)

Please allow 6-9 weeks for your first issue to arrive.

FOREIGN RATES

Payments must be in $US drawn on a US bank.

- ☐ $17 Canada/Mexico First Class
- ☐ $23 Rest of World Airmail
- ☐ $14 World Surface Mail

Delivery of foreign mail is slow and unreliable. We strongly advise airmail.

BACK ISSUES

$2.50 each:	$3.00 each:	
Vol. 6, Nos. 1, 2, 3, 4, 5	Vol. 7, Nos. 3, 4, 5, 6	Outside the U. S. add $ .50 per issue.
Vol. 7, Nos. 1, 2	Vol. 8, Nos. 1, 2	

FOREIGN DISTRIBUTORS OF *RECREATIONAL COMPUTING*

LP Enterprises, 313 Kingston Road, Ilford, 1G1 1PJ, Essex, UK; **Electronic Concepts PTY Ltd.**, 52-58 Clarence St., Sydney, NSW 2000, AUSTRALIA; **RS-232**, 186 Queen St., W. Toronto, Ontario M5V 1Z1, CANADA; **ASCII Publishing**, 305 HI TORiO, 5-6-7 Minami Aoyama, Minato-Ku, Tokyo 107, JAPAN; **Ing. W. Hofacker**, D815 Holzkirchen, Lindenstr. 8, WEST GERMANY; **Hobby Data**, S200 12 Malmo, SWEDEN; **Computer Store**, P. O. Box 31-261, 228 Milford Rd., Milford, Auckland, NEW ZEALAND; **Systems Formulate Corp.**, Shin-Makicho Bldg. 1-8-17, Yaesu, Chuo-Ku, Tokyo 103, JAPAN.

Printed by Nowels, Menlo Park, CA

Recreational Computing (ISSN #0164-5846) is published bimonthly by People's Computer Company, 1263 El Camino Real, Box E, Menlo Park, CA 94025. People's Computer Company is a tax-exempt, independent, non-profit corporation, and donations are tax-deductible. Second class postage paid at Menlo Park, California, and additional entry points.

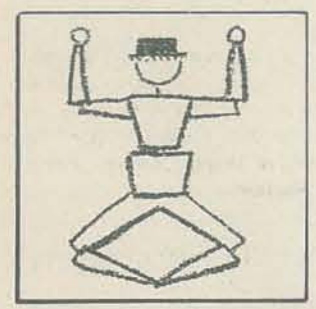
Recreational COMPUTING

Volume 8 Number 3
Nov-Dec 1979

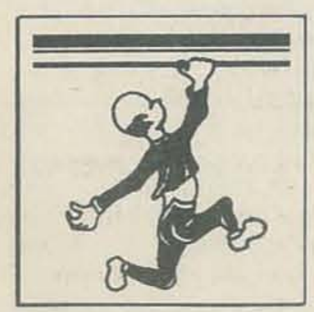
formerly people's computers

pg. 11

pg. 27



pg. 40



pg. 28

Cover by Ann Miya

Special Features

Articles

Games & Stuff

Departments

Editors' Notes

Who else but a Dragon could roast a hot dog with his breath? Well, here's the evidence.

Our cover shows the winning entry for the first International Computer-Drawn Dragon Contest, a fire-breathing but congenial scaly beast who evidently prefers roasting hot dogs rather than humans! It was drawn by Tom Weller of Berkeley, California using the Commodore PET graphic character set. Tom said that he typed all the graphics in "print" statements and that he "got pretty good at wrestling with the peculiar properties of the PET quote mode." Good work, Tom, you must have been determined! Tom's Dragon wins him a three-year subscription to RC.

The Second and Third place winning Dragons, drawn by two brothers, Steve and Tom Wuttke, aged 12 and 9, were also drawn on PETs. These Dragons were created as a part of the Second Kids Computer Clinic at the University of Wisconsin-Waukesha Campus. For their fine drawings, Steve wins either a two-year subscription or 12 back issues of RC *and Tom wins a one-year subscription or a copy of "What To Do After You Hit Return," P.C.C.'s first book of computer games.*

Fourth and Fifth place winners are John Livingston of Waukesha, Wisconsin, who also drew his Dragon on a PET, and Michael Slavin of Port Jefferson, New York, who drew his Dragon on a DEC PDP 11. They each get a consolation prize of a copy of P.C.C.'s Reference Guide to Personal and Home Computing. Thank you everyone who sent us Computer-Drawn Dragons. We really enjoyed all the contest entries! See page 7 for photos of the winners.

This issue is focusing on computer sound and music, since the sound feature seems to be the latest popular development in micros. The lead article, "See What You Hear and Hear What You See" by Herb Moore is the second in a series on the new Atari microcomputer. "The Joy of Sound" by Doc Plumber features a sound device that works on the TRS-80 and the PET. Alfred Bruey's article, "Making Music on the PET" describes a sound generation method for the machine. Also in the mode of expanding the capabilities of the micro is the article "Texas Instrument Graphics and Animation." The ideas and devices in these articles should bring hours of enjoyment for microcomputer users.

There are additional exciting items in these pages. Herb Kohl, a well-known educator, presents a challenge to programmers in a Shogi variation. FORTRAN Man has returned. Cryptarithms is still keeping our puzzlers busy. Also, we have included a number of listings or small programs throughout this issue, more than we have run in previous issues, since our readers have been requesting them.

We had mentioned last time that Don Inman might be added to the masthead of RC. *Well, we're happy to announce that it's confirmed. Don, a grizzled computer freak, prolific computer book writer, and former editor of* Calculators/Computers *will add bits of wisdom and insanity to the editorial staff of* RC. *Welcome, Don!*

Tracy Deliman   Don Inman
Ramon Zamora   Bob Albrecht

Letters

HELLO, COMPUTERTOWN!

Ok, ComputerTown, I've seen your announcement in several issues of *Recreational Computing* and have finally decided to write you and see what you are up to. First: some impressions after reading your announcement.

I assumed that something is going on or may go on to determine the future of P.C.C. Since P.C.C. is a non-profit corporation I would also assume you are wanting input (ideas, etc.) from those of us who help support P.C.C. with our subscriptions and donations.

All the labels probably tie in personal computers to your organization in present or future planning. Music in the park can be a nice thing—several people get together with some musicians and combine computer synthesized music with live music and you invite the public to participate and enjoy.

You take your computers to the local library, which is what I do, and share them with others less fortunate than yourself and do your part to help eradicate Computer Illiteracy.

Obviously you support or are being supported by several schools. I am interested in providing computer literacy courses to local schools who cannot afford to transport their students to the Lawrence Hall of Science.

I know nothing about Canada College. But, I do react to the fact that in our local junior college (Solano County) there is no course in micros.

I would also assume that the bookstore is important for I find myself and many other computer people to be book freaks.

As far as the Round Table pizza is concerned, in Fairfield, CA we meet every Wed. A.M. at Vickies restaurant to have coffee, breakfast and see what synergy we can get going.

Well, those are my thoughts, By the way, I and others in this area are very concerned over the future of personal computing and we have enjoyed your magazines and hope for more. If we can help in any way please feel free to contact us.

Tony Severa
Librarian, Midway Computer Club
131 Highland Ave.
Vacaville, CA 95683

Keep it up, Tony! You are right on target. We are out to make a small town of 27,000 people completely computer literate. Yes, we want input, ideas, and whatever. We see this as something that needs to be done—so we are doing it! Anyone needing info can send a SASE to ComputerTown, U.S.A.!, P.O. Box 310, Menlo Park, CA 94025 – Eds.

A READER QUIZZES RC

I am considering the purchase of a personal computer like the Apple II. Of those available this fall, the Atari 800 looks interesting. I plan to use my computer for personal use—education for my family, games and some personal finance. I plan on purchasing cassette programs for the most part in the $8-$20 range. I would also like to learn to do some of my own programming.

If I get the Apple II Plus with Applesoft II BASIC, do you anticipate that many of the programs for Apple that were integer BASIC would be available in Applesoft II?

Yes, as the users (you included) of Applesoft II make the conversions.

Is Applesoft II a more advanced BASIC than Integer BASIC?

Yes!

If you have seen or used the Atari 800, please write any observations below. I understand it is modular, holds up to 48K RAM, has color and sound capability with good graphics and is expandable. The ROM software and the 16K Atari RAM expansion prices are much higher than cassette software and 16K expansion kits for the Apple.

In your estimation, do you feel Atari will encourage outside software suppliers to sell low cost ($8-25) cassette software?

Yes, once the machines get into the "people's" hands.

Or sell low-cost memory upgrade kits?

Yes, from the regular hobby suppliers.

Do you (*RC*) have plans to have a number of articles on the Atari 800 when many are in use?

Yes! Lots of them. We are also running articles on the new TI 99/4. You saw it here first! We are doing articles now before there are many machines in use.

Any other comments on the Atari 800, etc.?

Send us an article when you get your machine.

Donald J. Stonek
6026 S. New York Ave.
Cudahy, Wisconsin 53110

SOMETHING'S BOTHERING HIM

There are two things that have been bothering me about the magazine the last issue or two. First and most important—what happened to SPOT? Where are all those neat little useful programs that were the cause of my subscribing in the first place?

The other thing on my mind is the Fantasy Games. It seems to me an exercise in futility to print long scenarios of non-existent games, fantasy, chess, tic-tac-toe, or any other kind. Where are the programs to go with them? How about a 4-D tic-tac-toe game with a 3-D board where pieces will at random pop off into a 4th dimension and pop back at some unpredictable time capturing whatever square they land on. How about a checker or chess game doing the same thing? If I were into games I probably could come up with something better. The point is that ideas for games are not very hard to come by (even for me); implementing the ideas turns out to be dragon feathers of a different kind.

Also, I see no point in reviews, commentary, or letters about movies, books and T.V. programs not directly related to computers in a computer magazine. I read *The Lord of the Ring* series but remember not one word about computers. Has my memory been attacked by a DELETE monster without my knowing it, or did I forget about the attack and then forget that I forgot?

Sincerely,
John W. Davison
P. O. Box 1165
Ft. Walton Beach, FL 32548

If you have any neat little programs for the PET, send them to the SPOT editor, Harry Saal. He needs them! Meantime, we'll publish more PET programs as readers send them in. – Eds.

USER CLUBS IN SO. CAL?

I perused your magazine at a local computer store and I liked what I saw.

Also, I am interested in joining a computer club. The Southern California Computer Society seems to have disappeared and I can find very little information on computer clubs. Does anyone have any information?

Donald C. Pickering
13425 Kornblum # E
Hawthorne, CA 90250

CRYPTIC FRIENDS

Dear Nine Hex,

I read and solved *most* of your cryptarithms in the latest *Recreational Computing.*

What intrigues me is your former association with the National Puzzlers' League. I do believe I recognize your nom, from the old issues of *The Enigma* that I have read, and am surprised and pleased to hear of you again.

The NPL, as you may know, fell on hard times during the 1960s and early 1970s. The group was close to dissolution on several occasions. I joined the League in 1973, and about that time there was a resurgence in interest and membership. Today we're a thriving organization again of 300 members, with more puzzles than ever, and resumed annual conventions.

Anyway, the note about you compelled me to write and say hello.

Will Shortz (Willz)
Associate Editor
Games Magazine

SIMULATION IN BUSINESS

The review of the simulation, "Starting a Small Business" by Leroy Finkel in your July-August issue inspired me to dig out my copies of the manuals for this simulation to see if it might be converted to run on our Alpha microcomputer system. I am pleased to report that after about 15 hours of effort, I was able to convert it to ALPHABASIC and to get it to run and print out successfully.

We plan to try the simulation in several of our business courses this fall. It will be a good way to introduce a touch of realism into our classes. We would be interested in learning about other business simulations—especially those already in BASIC!

Kurt Kreiselmaier, Assistant Professor
Dept. of Business Administration
Midwestern State University
3400 Taft Boulevard
Wichita Falls, TX 76308

REGARDING *UNIVERSE*

I am working on a program that is very similar to *Universe* (by Les La Zar in the March-April issue of *RC*). The difference is that I am writing it in Apple BASIC, using the Apple's high resolution graphics. Anyone who would like to help is welcome to contact me. I feel that if the program is to be any good, I should ask for help with it. After all I'm not a whiz at programming; you might say I'm just past novice.

My desire is to invite other people to join in writing a super fantastic program. Anyone contributing will hold the right to have their name listed as one of the authors. Also if you're interested in money, write me.

David Robertson
6338 Wisteria Lane
Apollo Beach, FL 33570

CHEERS FROM DOWN UNDER

Congratulations on a fine magazine. I find the games and chess discussions exactly to my taste.

Thanks for many happy hours of enjoyment.

Keith B. Lewis
Malvern, Victoria, Australia

TINY ZORK

The recent article you published on ZORK did little more than tempt us with the unobtainable. How many personal computer enthusiasts do you know that can read 9-track tape at all, let alone in DEC format?

Let's hope that you can come up with something that we can use, and publish it soon. And, please, remember that not all of us have TRS-80's either, even though it may sometimes seem so.

Thank you for your continuing good work.

Lewis Moseley, Jr.
2576 Glendale Court, N.E.
Conyers, GA 30208

Much the same comments have been expressed about Adventure . . . yet, we now have Adventure for the PET and TRS-80. As stated in the lead to the ZORK article, we want to promote a "Tiny Zork" for home computers. We are only tempting you with the obtainable! — *RZ*

TRACING GENEALOGIES

I bought a microcomputer last year, an Apple II (6502), in the hope that it would bring order out of chaos in the collected documentation I have of several thousand ancestors. I would like to be able to store, file, sort, retrieve, and cross-reference genealogical data, and be able to trace genetic disorders and make other analogs. I would like to have pedigree, individual and family group printouts as well as indexes.

The Mormons have done excellent work, but they use IBM 370's. Some work out of the University of Utah has focused on minis using an excellent soundex code with pointer systems for parents and progeny, but the adaptation to micros is not clear.

I would like to hear from others of a similar interest so that possibly a network of information could be pooled and shared.

Clifton M. Howard
58 Van Orden Road
Harrington Park, NJ 07640

CAN PETS SPEAK?

I am a recent subscriber to your magazine and I have been generally pleased with it. I would like to ask some things, mostly about PETs. I would very much like to be able to have my PET speak and/or recognize spoken words. Is there any way this can be done? I have been trying to get the PET to recognize words using both hardware and software together, but without much success. Also, do you think you could include more pertinent information and programs for the PET? Lately there hasn't been anything useful for the PET.

Did you know that the winning projects in the Southern California State Science Fair in ninth grade physical science and in ninth grade life science both had to do with having semi-artificial intelligence programs on PETs? This took place in Los Angeles during May. I think people who do projects like that deserve recognition.

Brian Sawyer
1310 Dover Hill Rd.
Santa Barbara, CA 93103

We are including more programs and information for PETs (check this issue) and would gladly include more if people would only send things! — *TD*

TOLKIEN DEBATE AGAIN!

It is a pity that the merits of various computers cannot be discussed without descending into disparagement of the good sense of those who chose otherwise than yourself—vide Mark T. Tsetsi in the July-August issue.

Even worse is the vitriolic condemnations by Lon Ponschock, in the same issue, of Tolkien's *Lord of the Rings*—without bothering to read it himself. Certainly it is escapist literature. It is a measure of Tolkien's artistry with words that some of us willingly suspend disbelief and escape into another, a magical world, with its own logic and development, one which answers some need deep within us. In this world love is not merely a matter of glands and hormones. The virtues and their opposites are not mere meaningless words or conditioned reflexes. Truth and beauty and deceit and ugliness exist and good and evil strive for mastery of the world.

It is not given to everyone to enter into Middle Earth. Even my wife and sons cannot follow and share completely its delights. I wish they could. I feel they are the poorer for not being able to do so. However, if I read his character correctly, I advise Lon Ponschock not to read Tolkien. *The Lord of the Rings* would surely, in his own words, offend him at every level at which he can be offended. It is not for him—a sad fact but true. There is enough unhappiness in the world without his adding to it by reading a work whose basic premises he will reject and whose development would fill him with revulsions. I suspect such an endeavor would so enrage him that another, and even worse, letter to *Recreational Computing* would inevitably result. Lon Ponschock is welcome to his literature. I hope he will have the courtesy to allow me to prefer mine.

Tolkien himself gave the last word on the subject. "Some who have read the book, or at any rate have reviewed it, have found it boring, absurd, or contemptible; and I have no cause to complain since I have similar opinions of their works, or of the kinds of writing that they evidently prefer."

For Lon Ponschock's edification, in Tolkien's own words, "As for any inner meaning or 'message,' it has in the intentions of the author none. It is neither allegorical nor topical."

As Sam said to Faramir, "It strikes me that folk takes their peril with them into Lorien, and finds it there because they've brought it." Any topical symbolisms which Lon Ponschock may find in *The Lord of the Rings* was brought there by him, not placed there by Tolkien, and reflects his feelings and beliefs, not those of the author.

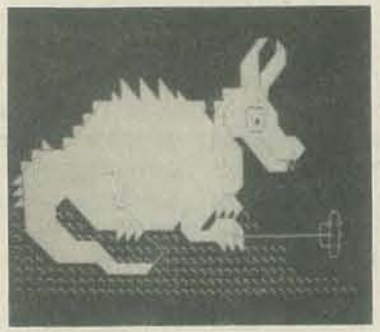
I subscribed to *Recreational Computing* to learn more about computers and programming. Let's stick to that subject and bring in other topics, no matter how worthy or delectable, only so far as they impinge upon it. That goes for puzzles, too.

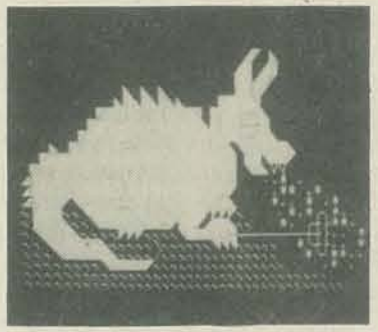
C. A. Cozart
440 S. 44th Street
Boulder, CO 80303

Remember, it is called Recreational Computing *not* Recreational Computers. *Computing can happen in the mind as well as in the computer. We blend the mind and the computer.* — *The Dragon*

Results of the First International Computer Drawn DRAGON CONTEST

First Place: Tom Weller (See the cover.)

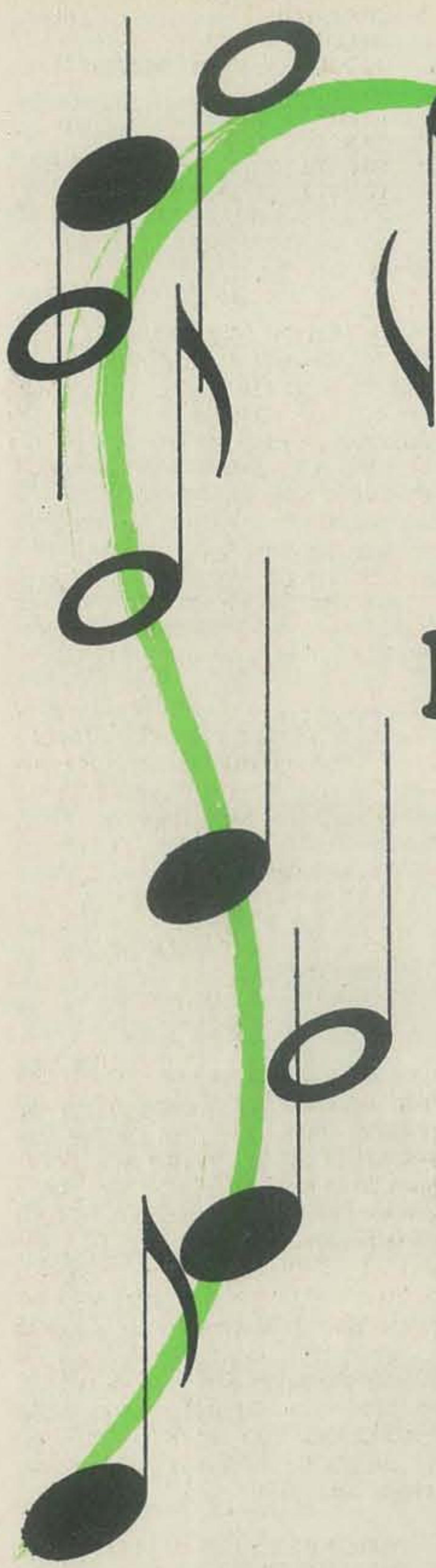


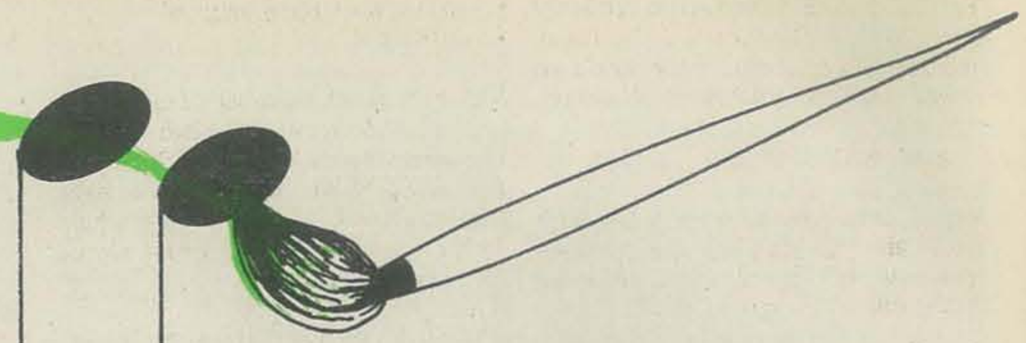


Second Place: Steve Wuttke

Third Place: Tom Wuttke

Fourth Place: John Livingston
Fifth Place: Michael Slavin





SEE WHAT YOU HEAR & HEAR WHAT YOU SEE

BY HERB MOORE

Our "music editor" is still busy exploring all the exciting dimensions of the new ATARI computers. Herb, who is a musician and beginning computer person, says that these articles, if read in sequence, can be understood by people who have had little previous experience in BASIC programming.

Herb's last article in the Sept-Oct issue of RC, covered the color graphics available on the machines while this one describes the sound capabilities. Later articles will combine color graphics and sound so that you really can "see what you hear and hear what you see." — *TD*

Sections of this article were excerpted from the book *ATARI BASIC* by Bob Albrecht, LeRoy Finkel, and Jerald R. Brown, ©1979, by John Wiley & Sons, Inc.

MAKING SOUNDS

Last issue you got to "see what you could see." This time let's "hear what you can hear."

As you might have guessed, but I'll tell you anyway, if you want the ATARI computer to make a sound, you need to enter a SOUND statement. If, however, your SOUND statement is "unsound" so to speak, the computer won't make any noises. That seems fair enough. So let's look more closely at what is necessary in the SOUND statement.

You need to include four variables: V (for *Voice*), N (for *Note*), T (for *Tone*), and L (for *Loudness*). These would be entered into the machine in this form:

```
20 SOUND V,N,T,L
```

If you enter the following program, it should give you your first chance to hear what you can hear and also allow you to set the volume on your TV set.

```
10 SOUND 0,121,10,8
20 GOTO 20
```

This will give you a steady tone that is approximately Middle C. The loudness should be at a medium range, so you can set the volume level on your TV set to a reasonable level while the tone is playing. Line 20 tells the computer to hold the tone until you press the BREAK key to stop the program.

The ATARI machine allows you to generate 16 different levels of loudness. If you wish to have a tone play at each level of loudness, you can use a FOR-NEXT loop as demonstrated in the following program:

```
10 FOR L = 0 TO 15
20 SOUND 0,115,10,L
30 FOR W = 1 TO 200: NEXT W
40 NEXT L
```

You now have a sense of the increments of loudness available, so let's set the loudness at an average level of 8 and look at the tone variable T. There are 16 tone settings on the ATARI. We can revise the above program using the variable T in the FOR-NEXT loop. The program will look like this:

```
10 FOR T = 0 TO 15
20 SOUND 0,115,T,8
30 FOR W = 1 TO 200: NEXT W
40 NEXT T
```

When this program is run, the computer will play each of the 16 available tones.

Next let's look at the variable N (for Note). The ATARI computer is able to select from 256 note values ranging from 0 through 255. Since we have 256 notes to produce, let's rewrite the program to vary the notes. We will also shorten the time delay in line 30.

```
10 FOR N = 0 TO 255
20 SOUND 0,N,10,8
30 FOR W = 1 TO 50: NEXT W
40 NEXT N
```

This will result in a set of notes that gradually decrease in pitch as the computer selects each new variable in the FOR-NEXT loop. That's right, it's "backwards"—lower numbers mean higher notes and higher numbers mean lower notes.

The chart gives numerical values for three octaves of notes corresponding to conventional tonality.

Natural Scale Note	N (Note) Value
C	243
C#	230
D	217
D#	204
E	193
F	182
F#	172
G	162
G#	153
A	144
A#	136
B	128
C (Middle C)	121
C#	114
D	108
D#	102
E	96
F	91
F#	85
G	81
G#	76
A	72
A#	68
B	64
C	60
C#	57
D	53
D#	50
E	47
F	45
F#	42
G	40
G#	37
A	35
A#	33
B	31
C	29

If you wish to write a musical scale, you can store notes from this chart in a table and read them into the program when you need them. Here's a program that plays two octaves of a C Major scale, ascending from Middle C. The C Major scale consists of all natural notes; that is, it has no sharps or flats.

```
10 FOR A = 1 TO 15
20 READ N
30 SOUND 0,N,10,8
40 FOR W = 1 TO 200: NEXT W
50 SOUND 0,0,10,8
60 FOR W = 1 TO 10: NEXT W
70 NEXT A
80 DATA 121,108,96,91,81
90 DATA 72,64,60,53,47
100 DATA 45,40,35,31,29
```

ATARI VOICES

So far so good. Now let's see if we can get the machine to "sing" to us in four part harmony. To do so, we'll need to explore the variable V (for Voice). The following program will make the machine play the three notes of a C Major chord, C, E, G, all three voices at once.

```
10 SOUND 0,121,10,8
20 SOUND 1,96,10,8
30 SOUND 2,81,10,8
40 FOR W = 1 TO 100: NEXT W
50 END
```

Now suppose you want to have voice 0 start and sustain a note, voice 1 coming on after a short delay and also sustaining, and finally voice 2 coming on to complete the chord. Try this:

```
10 SOUND 0,121,10,8
20 FOR W = 1 TO 100: NEXT W
30 SOUND 1,96,10,8
40 FOR W = 1 TO 100: NEXT W
50 SOUND 2,81,10,8
60 FOR W = 1 TO 100: NEXT W
70 END
```

As you typed in the same FOR-NEXT loop in lines 20, 40, and 60 in the program above, you may have said to yourself (if you're lazy like me), "There must be an easier way to do this." Or, if you've had a bit more programming experience, you may have said, "Why not use a SUBROUTINE here." A GOSUB statement tells the computer to leave temporarily the main program and go to a subroutine. In the subroutine, a RETURN statement tells the computer to continue with the main program at the line following the GOSUB statement. So we can rewrite and shorten our program using a subroutine:

```
10 SOUND 0,121,10,8: GOSUB 100
20 SOUND 1,96,10,8: GOSUB 100
30 SOUND 2,81,10,8: GOSUB 100
40 END
100 FOR W = 1 TO 100: NEXT W
110 END
```

OLE MAC WHAT?

Here's a slightly more complex program which uses subroutines to establish different possibilities for the duration of notes. Line 3000 will be equivalent to a quarter note, or a count of one beat in conventional music notation. Line 4000 will be equivalent to a half note, or two beats, line 5000 the equivalent of a dotted half note, three beats, and line 6000 will be the equivalent of a whole note or four beats.

```
 10 SOUND 0,243,10,8
 20 SOUND 1,193,10,8
 30 SOUND 2,162,10,8
 40 SOUND 3,60,10,11: GOSUB 5000
 50 SOUND 3,81,10,11: GOSUB 3000
 60 SOUND 0,182,10,8
 70 SOUND 1,144,10,8
 80 SOUND 2,121,10,8
 90 SOUND 3,72,10,11: GOSUB 4000
100 SOUND  0,243,10,8
110 SOUND 1,193,10,8
120 SOUND 2,162,10,8
130 SOUND 3,81,10,11: GOSUB 4000
140 SOUND 3,47,10,11: GOSUB 4000
150 SOUND 0,162,10,8
160 SOUND 1, 128,10,8
170 SOUND 2,108,10,8
180 SOUND 3,53,10,11: GOSUB 4000
190 SOUND 0,243,10,8
200 SOUND 1,193,10,8
210 SOUND 2,162,10,8
220 SOUND 3,60,10,11: GOSUB 6000
230 END

3000 FOR W = 1 TO 100: NEXT W
3500 RETURN
4000 FOR W = 1 TO 200: NEXT W
4500 RETURN
5000 FOR W = 1 TO 300: NEXT W
5500 RETURN
6000 FOR W = 1 to 400: NEXT W
6500 RETURN
```

You may notice that Voice 3 in this program is set at a slightly higher level of loudness. Voice 3 is the melody voice. Voices 0, 1, and 2 play a background chord so are set lower. If you run this program, you might find the melody vaguely familiar. That's because the melody is *almost* "Ole MacDonald Had a Farm." But . . . not quite!

The machine will execute commands so fast that there will be no space between the notes, causing them to slur together. If we rewrite the program and put a time delay in the subroutines, we can distinguish each separate note in the melody:

```
3000 FOR W = 1 TO 100: NEXT W
3100 SOUND 3,0,0,0
3200 FOR W = 1 TO 10: NEXT W
3500 RETURN
4000 FOR W = 1 TO 200: NEXT W
4100 SOUND 3,0,0,0
4200 FOR W = 1 TO 10: NEXT W
4300 RETURN
5000 FOR W = 1 TO 300: NEXT W
5100 SOUND 3,0,0,0
5200 FOR W = 1 TO 10: NEXT W
5300 RETURN
6000 FOR W = 1 TO 400: NEXT W
6500 RETURN
```

The statements in lines 3100-3200, 4100-4200, 5100-5200 tell the machine to turn off Voice 3 for a count of 10. This delay will be short since the computer counts quite fast. There is no short time delay in subroutine 6000. All the voices are turned off by the END statement in line 230.

MORE IMPROVEMENTS

In the main text of the program, we change line 40 to these three lines:

```
40 SOUND 3,60,10,11: GOSUB 3000
41 SOUND 3,60,10,11: GOSUB 3000
42 SOUND 3,60,10,11: GOSUB 3000
```

Instead of holding the same note for 3 beats, we're telling the machine to stop briefly after playing the note for one beat, then repeat this process two more times so that we get three distinct notes. These lines are equivalent to striking the key on a piano three separate times versus striking the key once and holding it down for three beats. We make similar changes as follows:

```
 90 SOUND 3,72,10,11: GOSUB 3000
 91 SOUND 3,72,10,11: GOSUB 3000
140 SOUND 3,47,10,11: GOSUB 3000
141 SOUND 3,47,10,11: GOSUB 3000
180 SOUND 3,53,10,11: GOSUB 3000
181 SOUND 3,53,10,11: GOSUB 3000
```

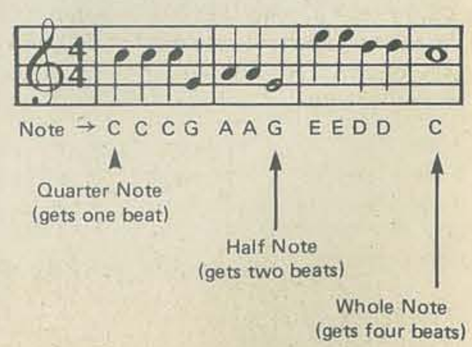
We'll leave line 130 alone since it is meant to be a half note that is sustained 2 beats.

NOW HEAR THIS!

You could say that part of the "Zen in the Art of Programming" is finding different ways to make the machine do what you want it to. So for the zen of it, here's another way to have the machine play the same tune. We have made the SOUND statements into subroutines and will enter different values for the note variables in the program. The variable Y in line 7000 of the FOR-NEXT loop allows you to vary the note duration for voice three. Lines 8000 through 8500 create the time delay that allows you to turn off the sound when you wish to separate one note from the next. Here is a duet of "Ole MacDonald" and subroutines:

```
  10 N0 = 243: N1 = 193: N2 = 162: GOSUB 6000
  20 N3 = 60: Y = 100: GOSUB 6400
  30 N3 = 60: Y = 100: GOSUB 6400
  40 N3 = 60: Y = 100: GOSUB 6400
  50 N3 = 81: Y = 100: GOSUB 6400
  60 GOSUB 8000
  70 N0 = 182: N1 = 144: N2 = 121: GOSUB 6000
  80 N3 = 72: Y = 100: GOSUB 6400
  90 N3 = 72: Y = 100: GOSUB 6400
 100 GOSUB 8000
 110 N0 = 243: N1 = 193: N2 = 162: GOSUB 6000
 120 N3 = 81: Y = 200: GOSUB 6400
 130 N3 = 47: Y = 100: GOSUB 6400
 140 N3 = 47: Y = 100: GOSUB 6400
 150 GOSUB 8000
 160 N0 = 162: N1 = 128: N2 = 108: GOSUB 6000
 170 N3 = 54: Y = 100: GOSUB 6400
 180 N3 = 53: Y = 100: GOSUB 6400
 190 GOSUB 8000
 200 N0 = 243: N1 = 193: N2 = 162: GOSUB 6000
 210 N3 = 60: Y = 400: GOSUB 6400
 220 GOSUB 8000
6000 SOUND 0,N0,10,8
6100 SOUND 1,N1,10,8
6200 SOUND 2,N2,10,8
6300 RETURN
6400 SOUND 3,N3,10,11
7000 FOR W = 1 TO Y: GOSUB 8300
7100 RETURN
8000 SOUND 0,0,0,0
8100 SOUND 1,0,0,0
8200 SOUND 2,0,0,0
8300 SOUND 3,0,0,0
8400 FOR Z = 1 TO 10: NEXT Z
8500 RETURN
```

Now you have an idea of some of the things that you can do with the sound parameters of the ATARI computer. In this article I have dealt with the voice and note parameters of the SOUND routine. You might want to now play with the T (for *T*one) and L (for *L*oudness) to further enrich your music programs. I'll be interested to hear what you hear. For your reading pleasure, I'm including a description of "Ole MacDonald" in conventional music notation. Be "hear-ing" you!



MATCH ME

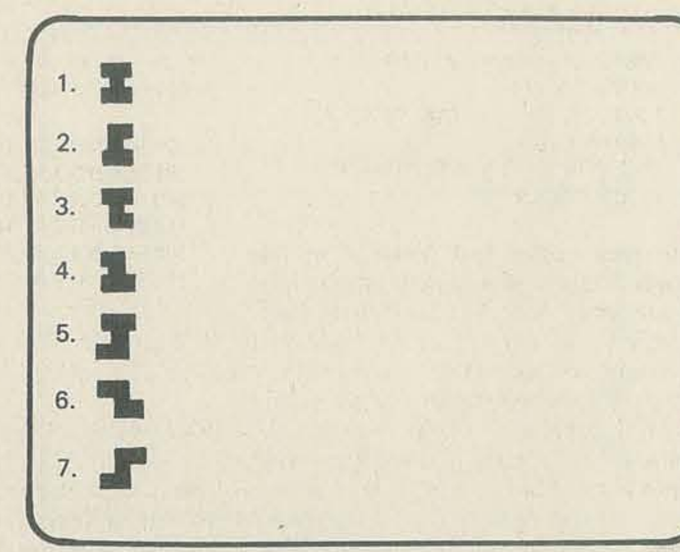
*Don Inman's son, Kurt, is already quite proficient on the TRS-80. He and Don are co-authors on several books that deal with this fast-selling machine (*Introduction to T-BUG *and* Real Time BASIC for the TRS-80, *dilithium Press). Kurt is fifteen years old!*

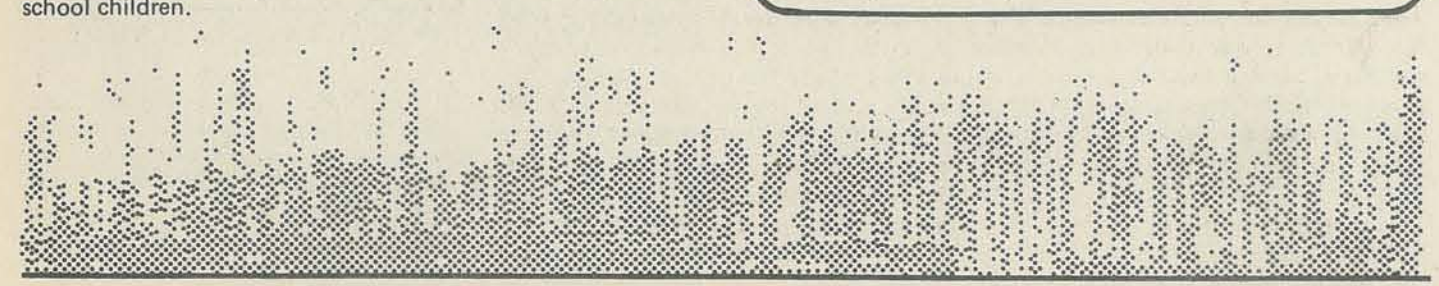
But how about Don's daughter, Kit? She is too young to co-author books, right now. However, she can run the computer. Don has produced a shape-matching program for her which he now shares with you and your kids. —*RZ*

BY DON INMAN

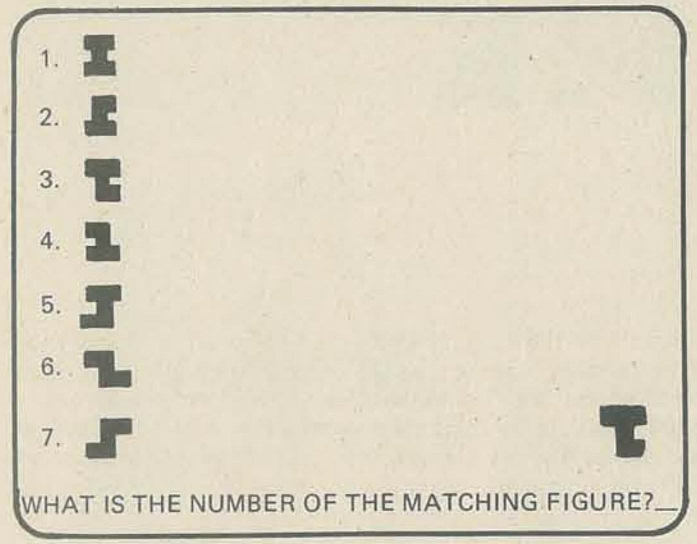
The use of graphics on the computer is the key to capturing the interest of youngsters. This program was designed for my daughter, Kit, who is in kindergarten. It requires that the user be able to punch in a single-digit number (1-7) and the ENTER key on the TRS-80 computer. It could also be used by pre-school children.

Seven shapes are drawn on the left edge of the video screen.

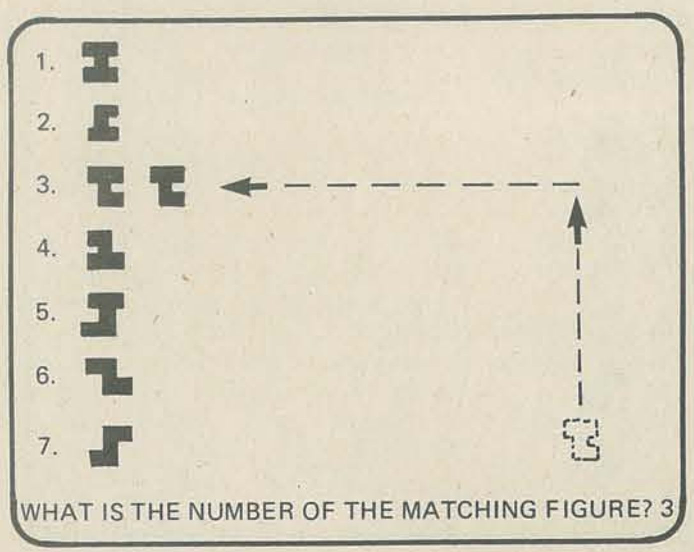




One of these seven shapes is then randomly chosen and it appears in the lower right corner of the screen along with the question, "WHAT IS THE NUMBER OF THE MATCHING FIGURE?"



The user then keys in the number of the shape which he/she thinks the random shape matches. The ENTER key is then pressed. The random shape then moves vertically and horizontally to a position adjacent to the chosen shape.



A message then tells whether the choice was correct or not.

My daughter is able to match the shape 100% of the time. After such success, the level of difficulty should be increased. This can be done by adding a method of timing the responses or by making the shapes more complex.

EXPLANATION OF THE PROGRAM BY SECTIONS

Variables:

S is used to record the number of correct guesses.
K is used to record the number of incorrect guesses.
B is a random number (one of the seven numbered shapes).
P, L are decimal numbered memory locations for original shapes.
M,N are graphic character codes.
Z is the chosen input number (to match the shape).
C is used in the shape move to select memory locations.
PLOT is an array used to select the location and figures of the moving shapes.

In this first section, line 40 initializes variables. Line 50 clears the video screen and reads in the data for PLOT which is used to move the random shape when a choice has been made for matching. Line 60 reads in data and plots the seven shapes. Line 70 numbers the seven shapes. Line 80 selects a random number (1-7) which will determine which shape must be matched. Line 90 sets an initial value for P.

```
40 S=0: K=0: DIM PLOT(7,3)
50 CLS: FOR A=1 TO 7: FOR B=1 TO 3: READ PLOT(A,B): NEXT B,A
60 FOR A=1 TO 14: READ P,N: POKE P,N: NEXT A
70 FOR A=1 TO 7: READ L,M: PRINT @L,M;: NEXT A
80 B= RND(7)
90 P = 16251
```

In the next section, a graphic character code is selected for N and M depending on the value of the random number. This selects the correct shape to match the corresponding shape numbered by the value of B. After the correct values for N and M have been chosen, the program proceeds to line 200.

```
100 IF B=1 THEN N=187: M=183
110 IF B=2 THEN N=186: M=183
120 IF B=3 THEN N=171: M=183
130 IF B=4 THEN N=187: M=181
140 IF B=5 THEN N=187: M=151
150 IF B=6 THEN N=171: M=181
160 IF B=7 THEN N=186: M=151
```

Next we put the random figure on the screen and get the guess, Z. For a correct guess Z will be the same value as B (the number selected randomly by the computer). Line 200 displays the random shape made up of the character codes N and M. Line 210 asks for the user to make his guess. Line 220 makes sure that the number selected is either 1,2,3,4, 5,6,or 7. If the number chosen is not one of these, the selection is erased and the program goes back for a new input. Line 230 causes the program to omit any vertical movement, as the guess is already positioned on the correct line.

```
200 POKE P,N: POKE P+1,M
210 PRINT @906,"WHAT IS THE NUMBER OF THE MATCHING FIGURE";: INPUT Z$
215 Z=VAL (Z$)
220 IF Z<>1 AND Z<>2 AND Z<>3 AND Z<>4 AND Z<>5 AND Z<>6 AND Z<>7
    PRINT @906,;CHR$(30): GOTO 210
230 IF Z=7 GOTO 400
```

The vertical movement is controlled by lines 300 and 310. This is a loop which erases the random figure at its current position (line 300), and then replots it on the next line up (line 310). This continues until the correct horizontal line has been reached (determined by the value input for Z and selected by PLOT (Z,1)). The inner loop (FOR W – NEXT W) is merely a time delay (which may be varied).


```
300 FOR C=16251 to PLOT(Z,1) STEP-64: POKE C,32: POKE C+1,32
310 POKE C-64,N: POKE C-63,M: FOR W= 1 to 25: NEXT W,C
```

The horizontal movement of the random shape is done next. A loop similar to that of the vertical movement is used. Line 400 wipes out the figure, and line 410 replots the new figure. A time delay is used again.

```
400 FOR C=PLOT(Z,2) to PLOT(Z,3) STEP-4: POKE C,32: POKE C+1,32
410 POKE C-4,N: POKE C-3,M: FOR W=1 TO 25: NEXT W,C
```

We are now ready to print a message which tells the user whether or not the guess was correct. If the guess is incorrect, line 500 causes the program to skip to line 600. If the guess is correct, line 510 prints the message, and line 520 reads data and prints a smiling face along with the number of correct and incorrect guesses. Line 530 then causes the program to skip over the incorrect message and go to line 700.

```
500 IF B  Z GOTO 600
510 PRINT @90," A MATCH":
520 FOR A=1 TO 40 READ B,C; POKE B,C: NEXT A
530 GOTO 700
```

The next section contains the message for an incorrect guess. Line 600 increases the count for number of misses. Line 610 and 620 print the message. Line 630 is a time delay to allow the message to be read. Lines 640 through 660 erase the message and the shape, and line 670 returns to line 90 to put the random shape in the lower right corner ready for a new guess.

```
600 K=K+1
610 PRINT @276,"NO,NO - THAT'S A MISS!";
620 PRINT @404,"THINK HARD, THEN TRY AGAIN.";
630 FOR W=1 TO 750: NEXT W
640 POKE C,32; POKE C+1,32
650 PRINT @276,"                    ";
660 PRINT @404,"                    ";
670 GOTO 90
```

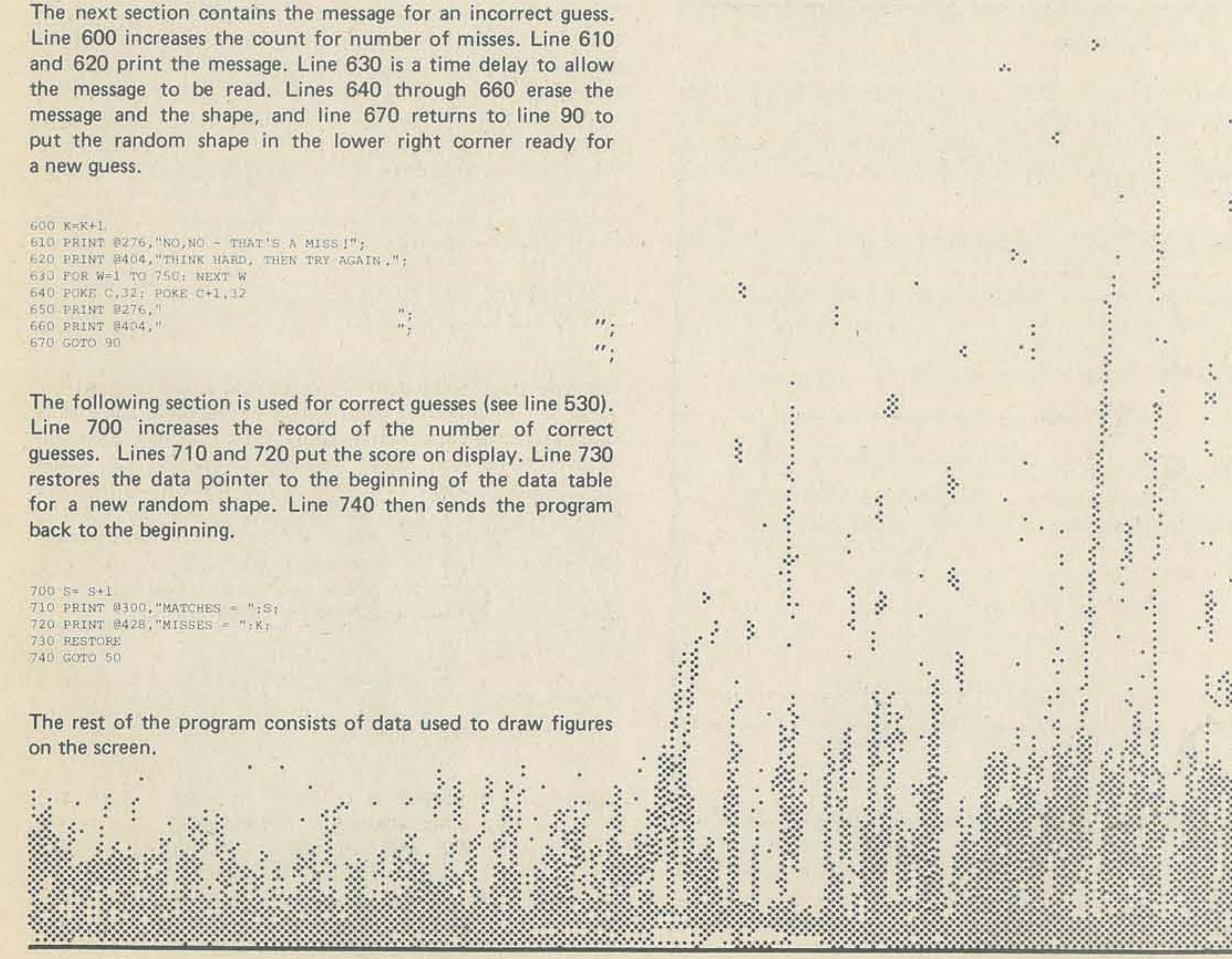
The following section is used for correct guesses (see line 530). Line 700 increases the record of the number of correct guesses. Lines 710 and 720 put the score on display. Line 730 restores the data pointer to the beginning of the data table for a new random shape. Line 740 then sends the program back to the beginning.

```
700 S= S+1
710 PRINT @300,"MATCHES = ";S;
720 PRINT @428,"MISSES = ";K;
730 RESTORE
740 GOTO 50
```

The rest of the program consists of data used to draw figures on the screen.

```
1000 DATA 15547,15483,15432
1010 DATA 15675,15611,15560
1020 DATA 15803,15739,15688
1030 DATA 15931,15867,15816
1040 DATA 16059,15995,15944
1050 DATA 16187,16123,16072
1060 DATA 16251,16251,16200

1100 DATA 15428,187,15429,183,15556,186,15557,183
1110 DATA 15684,171,15685,183,15812,187,15813,181
1120 DATA 15940,187,15941,151,16068,171,16069,181
1130 DATA 16196,186,16197,151

1200 DATA 65,1,193,2,321,3,449,4,577,5,705,6,833,7

1300 DATA 15578,152,15579,131,15580,131,15581,131
1310 DATA 15582,131,15583,131,15584,131,15585,164
1320 DATA 15640,168,15641,174,15642,32,15643,32
1330 DATA 15644,140,15645,160,15646,144,15647,140
1340 DATA 15648,32,15649,32,15650,157,15651,148
1350 DATA 15704,130,15705,171,15706,32,15707,136
1360 DATA 15708,176,15709,178,15710,177,15711,176
1370 DATA 15712,132,15713,32,15714,151,15715,129
1380 DATA 15770,137,15771,176,15772,178,15773,179
1390 DATA 15774,179,15775,177,15776,176,15777,134
```

Lines 1000-1060 fill the PLOT array. The seven shapes and their memory locations are stored in lines 1100-1130. The numbers for the shapes are taken from line 1200. Lines 1300-1390 are the data used to draw the smiling face. Write and let us know what you and your children's experience with this program is like.



A Shogi Variation

BY HERBERT KOHL

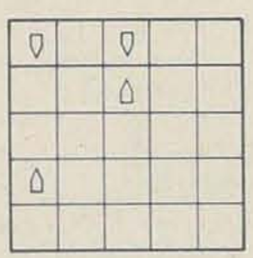
Herb, a well known educator, author, and editor, inaugurates a new RC challenge series: games for you to program. You can tell from this first offering that the games you will see here will be interesting. (They will also be educational–but we'll try to keep that quiet!)

Herb and his wife Judith are co-directors of Coastal Ridge Research and Education Center in Pt. Arena, CA. The center is a non-profit educational organization designed to provide both theoretical and practical support to people involved in progressive educational activities.

Let us know how you like this series. – *RZ*

Shogi is the Japanese version of chess. Though in many ways it is similar to chess it has one feature that makes it unique among chess type games. All of the pieces in the game, no matter what rank, are shaped like arrows:

The pieces are also all the same color. The only way you can tell which player controls a piece is by the direction the arrow is pointing. Your pieces point towards your opponent.



During the game of Shogi, any captured piece can be put back into play. The person who captures the piece can replace it on the board in lieu of one of his or her moves. The piece then becomes part of the player's forces.

Because of this characteristic, the game strategy gets complex. You must analyze not only the situation on the board, but the off-the-board power your opponent has.

The rules of Shogi are also complex and will be described in a future article. Here I would like to demonstrate the rule of interchangeability of pieces on a simplified board. To get a feel for the consequences of the rule, a 5 X 5 board is used for experimentation. In the games I've begun to explore, each player starts with 5 pieces, a ruler and four warriors. The rulers move as kings do in chess, and as in chess they cannot move into check. The rulers have no capturing moves. The warriors have modified pawn moves. They move and capture like pawns with these exceptions:

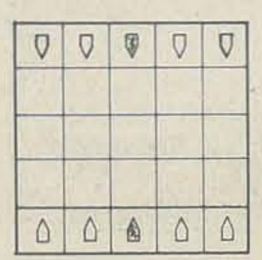
1) They cannot move 2 squares on their first move.
2) They can move one square at a time backwards as well as forwards.
3) There is no pawn promotion when a piece reaches the last rank.

The object of the game is to either:

1) Capture the ruler.
2) Trap your opponent so that none of his or her pieces can make a legitimate move.

There is one additonal rule–the Shogi rule. You can put a warrier you capture back into play on your side anywhere on the board instead of moving one of your pieces.

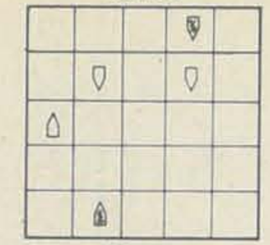
At the beginning of the game the board looks like this:



Later in the game, here is a possible sequence of moves using the Shogi rules.

The board looks like this. It is South's turn to move.

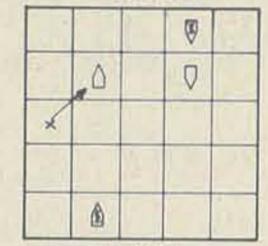
NORTH



SOUTH

A South warrior captures a North warrior.

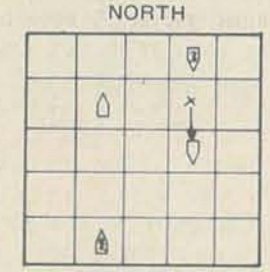
NORTH



SOUTH

A North warrior advances.

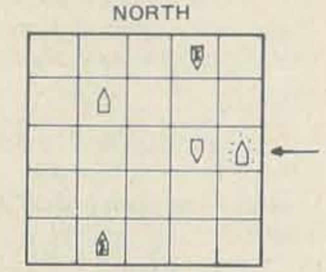
NORTH



SOUTH

South puts a captured warrior into play. Trouble for North!

NORTH

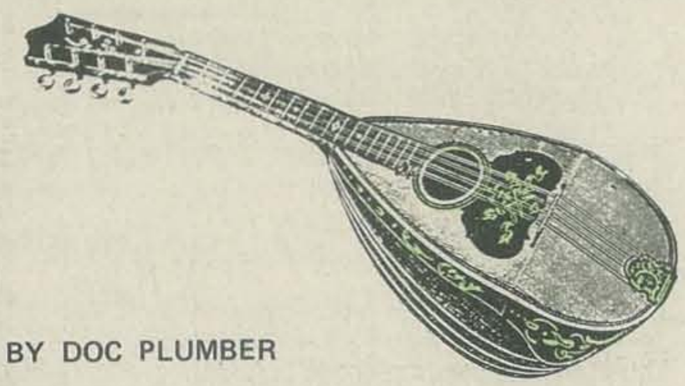


SOUTH

Is it possible to program this game and explore the possibilities for interesting play? The board can be expanded. New pieces can be added that move like bishops or rooks to further complicate the game. What would a game exploration program look like? Could there be a way to develop a program so that the size of the board and the move characteristics of the piece can be varied each time one plays?

The Joy of Sound from SOUNDWARE

(Press ENTER to Continue)



BY DOC PLUMBER

He's back again, this time with a system that allows you and your TRS-80 to make music together! This is the second in a series by Doc Plumber on hardware/software interfaces that even amateurs can put together.

The next step is to bring several computers together and create a symphony! Or would it be a cacophony? – *TD*

If you own a TRS-80, PET or COMPUCOLOR II, then SOUNDWARE™* from CAP Electronics is for *you*. This simple-to-connect, easy-to-use system will give you hours of entertainment when used with the included demonstration program alone. If you add its capabilities to your current game programs, you'll get realistic sounds that will make your programs come alive.

SOUNDWARE is a combination hardware/software package (the software package can be purchased separately) designed for the PET and TRS-80.

*TM a trademark of CAP Electronics, 1884 Shulman Ave., San Jose, CA 95124 – $29.95 for the PET and $39.95 for the TRS-80.

HARDWARE

The hardware installation is easy. The package is about the size of a small transistor radio. When you open the case to insert two AA batteries, you'll find what looks suspiciously like the audio section of a transistor radio. Clever idea! (Why didn't I think of it?)

Next, you merely unplug the AUXiliary connector of the cassette recorder (the gray 3.5 mm plug to your Radio Shack cassette) and plug it into the SOUNDWARE connector. An On/Off/Volume control is provided to adjust the volume of the sound.

That's all there is to the hardware. No tools, wiring or soldering are necessary.

SOFTWARE

Three programs are provided on the SOUNDWARE cassette tape. The first is a demonstration program that is loaded by the TRS-80 command:

CLOAD "A"

After this portion has been loaded, you see on the screen the message:

WORKING

After a few seconds the screen displays the title shown at the heading of this article. When the ENTER key is pressed, the menu for the demonstration is printed.


```
SOUNDWARE DEMONSTRATION PACKAGE

A – SAUCER
B – LITTLE LASER
C – BIG LASER
D – BOMB
E – LITTLE BIRDS                 SOUNDWARE
F – BIG BIRDS
G – CHIPMUNKS                       BY
H – PONG                      CAP ELECTRONICS
I – THEIR POLICE
J – OUR POLICE
K – MACHINE TALK
L – BACK TALK
Q – QUIT                  CHOOSE YOUR SOUND !!!>_
```

By pressing any one of the keys (A-L), you will hear the sound of your choice. If you goof and press the wrong key (one not on the menu), the screen will display the title "page" and you must once again press ENTER to see the menu.

After listening to this wide variety of demonstration sounds, you will be ready to experiment with some sounds of your own. If you press Q, control is return to BASIC with the usual prompt:

READY
>_

You are now ready to load the next program on the SOUNDWARE tape. Type:

CLOAD "B"

```
SOUNDWARE SOUND COMPOSER

ENTER: PITCH, DURATION, #STEPS, STEPSIZE1, STEP-
       SIZE2
       (1-999), (1-9999), (0-255), (0-255), (0-255)

EXAMPLES: 1-TONE     100, 500, 0, 0, 0
          2-LASER    5, 5, 100, 2, 0
          3-SAUCER   250, 25, 249, 5, 248

PRESS: 1 OR 2 OR 3 TO HEAR AN EXAMPLE
       E TO ENTER A NEW SOUND
       R TO REPEAT THE PREVIOUS SOUND
       Q TO QUIT
```

The display is self-explanatory. Pressing 1, 2 or 3 will give you one of the three sounds listed as examples. If you want to enter a new sound, type:

E

Below the instructions in the previous display will appear:

```
Q TO QUIT

FR, DU, #S, SS1, SS2 = ? _
```

You now type in your five parameters (separated by commas). Example:

FR, DU, #S, SS1, SS2 = ? 5, 5, 100, 2, 20
then press: ENTER
WOW!!! FAST FIRING LASERS!!!

Notice the tone example uses the two parameters 100 and 500 followed by three zeros. This plays a single tone with a pitch value of 100 and a duration value of 500. Let's experiment by lengthening the duration. It can go up to 9999.)

Type: E
then the parameters: 100, 1000, 0, 0, 0 ← this is doubled
then press: ENTER

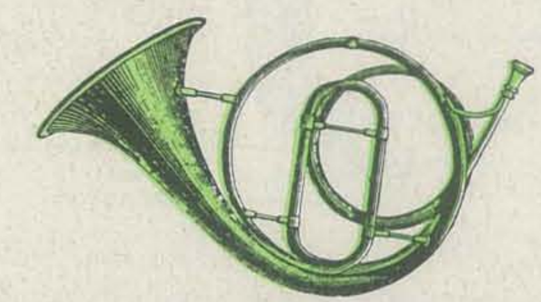
```
THE NOTE LASTED TWICE AS LONG AS THE ORIGINAL
```

Now type: E
then: 200, 500, 0, 0, 0 (500 ← original duration; 200 ← pitch is doubled)
then press: ENTER

```
THE NOTE IS LOWER IN PITCH THAN THE ORIGINAL
```

Now type: E
then: 50, 500, 0, 0, 0 (50 ← pitch is half original)
then press: ENTER

```
THE NOTE IS HIGHER IN PITCH THAN THE ORIGINAL
```



Thus, we see that low values produce high pitches, and high values produce low pitches. High values produce long durations, and low values produce short durations. You may have also notice that the high pitched note was shorter than either of the other two even though the durations were the same (500). To have a true constant duration, the product of the pitch and the duration values should be kept constant. Example:

original note: 100 X 500 = 50,000
high pitch: 50 X 1000 = 50,000
low pitch: 200 X 250 = 50,000

Therefore, if we enter:

100,500,0, 0, 0,
or 50,1000, 0, 0, 0
or 200, 250, 0, 0, 0

notes of equal duration should be produced.

Next, we'll examine how the steps work.

1. Original note plus five steps of size 6.

```
Type: E
      100, 500, 5, 6, 6
                      ↖ ─────────── each step size 6
                    ↖ ───────────── 5 additional steps
```

We hear:
original
step down #1
step down #2
step down #1
step down #2
step down #1

Down we go, alternating steps 1 and 2 from the original note.

2. Original note plus five steps of size 250.

```
Type: E
      100, 500, 5, 250, 250
                      ↖ ─────── each step size 250
```

We hear:
step up #1
step up #2
step up #1
step up #2
step up #1
original note

3. Alternating up and down

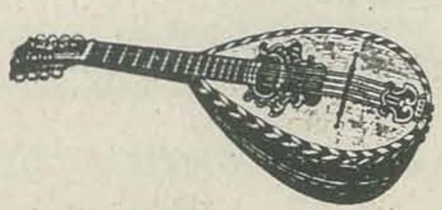
```
Type: E
      100, 500, 20, 250, 6
                         ↖ ──────── step down
                    ↖ ───────────── step up
                ↖ ───────────────── 20 additional steps
```

We hear:

. . . etc.

4. One last experiment varying the duration with alternating steps

Type: E	20, 20, 250, 250, 6	sounds an alarm
E	20, 200, 250, 250, 6	gives a twinkle
E	20, 2000, 50, 250, 6	HELP! the police



These experiments could go on for hours as there are so many possible combinations of the parameters.

Each step value is added (or subtracted) to the previous value, but some puzzling results can occur when you use more than one step. If you take only one step:

```
100, 500, 1, 3, 0
              ↖ ──────── small numbers give
                         a small step down

100, 500, 1, 170, 0
                ↖ ────── larger numbers give
                         larger steps down

100, 500, 1, 250, 0
                ↖ ────── very large numbers give
                         a small step up
```

To clear up the picture a little bit, study the following table of results.

Step	Change
255	No change
254	Small step up
↕	↑ to ↓
157	big step up
156	stay away
155	big step down
↕	↑ to ↓
1	small step down
0	no change

When more than one step is involved and large steps are used, you may get a "wrap around" from high to low or from low to high pitches.

USING SOUNDWARE IN PROGRAMS

The third program on the SOUNDWARE tape is helpful in creating sounds in your own programs. You first load SOUNDWARE Program "C" into the computer from the cassette. It is a BASIC program which will load the machine language program used to make the sounds. When it has finished loading, you will receive the usual READY signal from the TRS-80. Here is how the screen looks from power up.

```
MEMORY SIZE? 32642          saves room for the machine
                            language program
RADIO SHACK LEVEL II BASIC
READY
>CLOAD "C"                  SOUNDWARE loaded here
READY
>_
```

Now type:

RUN: Run the SOUNDWARE "C" program which loads the machine language program for you.

The screen goes blank, and then displays:

```
MEMORY SIZE MUST BE SET TO AT LEAST 125
BYTES LESS THAN MAX E.G. 32542 FOR 16K OR
20324 FOR 4K MACHINES
IF IT'S NOT, POWER OFF AND START AGAIN!!

PRESS ENTER TO CONTINUE_
```

Now press:

ENTER

```
MEMORY SIZE MUST BE SET TO AT LEAST 125
BYTES LESS THAN MAX E.G. 32642 FOR 16K OR
20324 FOR 4K MACHINES
IF IT'S NOT, POWER OFF AND START AGAIN!!

PRESS ENTER TO CONTINUE
WORKING                       Delay here a few seconds to
STARTING ADDRESS = 32642      load machine language
                              program
NOW YOU COULD MAKE SOUNDS WITH X = USR(0)

THE SOUND PARAMETERS CAN BE LOADED INTO
TM TO TM+6 WITH POKES
READY
>_
```

The computer is now ready for you to enter your BASIC program. Your program can be entered from the keyboard or from a pre-recorded cassette tape. Since Don Inman was kind enough to send me a cassette copy of MATCH ME (see page), I decided to add some sound to his program.

My next step was to load MATCH ME from the cassette.

I typed:

CLOAD "MATCH ME"

After the program had been entered, I made these changes and additions to MATCH ME.

SUBROUTINE ADDED TO PLAY SOUND

```
2000 TM = PEEK (16561) + PEEK (16562) * 256 + 2
2010 POKE TM,D: POKE TM + 2,E: POKE TM + 4,F:
     POKE TM + 5,G: POKE TM + 6,H
2020 X = USR (0): RETURN
```

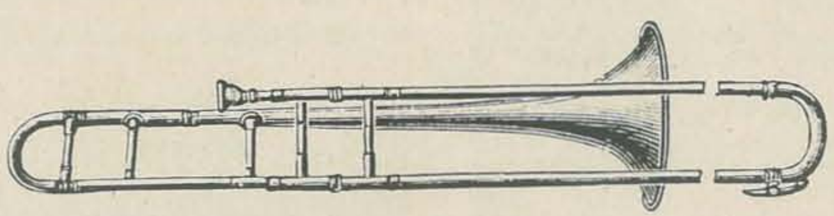
MAIN PROGRAM CHANGES AND ADDITIONS

Line	Statement	Effect
55	D = 200: E = 5: F = 100: G = 255: H = 0: GOSUB 2000	Produces a sound similar to a siren on its way up. Used to introduce the program and after each correct match.
70	FOR W = 1 TO 7: READ L , M: PRINT @ L , M	Provides a laser shot as each of the figures are numbered.
75	D = 15: E = 10: F = 30: G = 1: H = 0: GOSUB 2000: NEXT W	
205	D = 20: E = 100: F = 0: G = 0: H = 0: GOSUB 2000	Adds a short tone (or blip) when the figure to be matched appears.
310	POKE C−64 , N: POKE C−63 , M	
320	D = 20: E = 100: F = 0: G = 0: H = 0: GOSUB 2000	Produces a short tone as the figure moves vertically.
410	POKE C−4 , N: POKE C−3 , M	Produces a different tone as the figure moves horizontally.
420	D = 40: E = 50: F = 0: G = 0: H = 0: GOSUB 2000	
515	D = 10: E = 100: F = 20: G = 255: H = 6: GOSUB 2000	Produces falling notes after A MATCH is printed.
525	D = 10: E = 100: F = 20: G = 6: H = 250: GOSUB 2000	Produces a series of monotones after the smiling face.
615	D = 250: E = 10: F = 50: G = 5: H = 249: GOSUB 2000	Produces a rising series of warbling notes after NO, NO . . . on a missed guess.
630	D = 150: E = 10: F = 90: G = 249: H = 9: GOSUB 2000	Produces a falling series of warbling notes after TRY AGAIN . . .
715	D = 50: E = 40: F = 20: G = 250: H = 6: GOSUB 2000	Produces a series of low monotones after the number of matches are printed.
725	D = 30: E = 67: F = 20: G = 245: H = 11: GOSUB 2000	Produces a series of higher monotones after the number of misses are printed.
740	FOR W = 1 TO 30: D = RND (255): E = 50: F = 0: G = 0: H = 0: GOSUB 2000	Produces 30 random tones before proceeding to a new try.
745	NEXT W	

Whenever sound is needed in the main program, the parameters (D, E, F, G and H) are assigned and the subroutine at line 2000 is called. A variety of sounds were added to the program, but I'm sure that you could do so much better. Here is a brief description of the sounds I used. You'd have to play them on SOUNDWARE to appreciate the effect.

MY RECOMMENDATION ON SOUNDWARE

Get it as soon as possible! You don't know what you've been missing. The sounds that you can create are endless. Let your imagination run wild! Be careful if you have a room full of computers equipped with SOUNDWARE. You may wish you had a set of earphones for each machine. SOUNDWARE is addictive. Once you start using it, you'll have trouble stopping.



Cryptarithms

BY JOHN DAVENPORT CREHORE (Nine Hex)

As many of you discovered, Puzzle 12 in the last issue had an error in it. The puzzle should have read: Puzzle 12 (Computers)

```
 L SUL TAL
 E SSL TTY
BM MHH HHH
```

We apologize for the frustrations this may have caused.

Look elsewhere in this issue for some example programs that readers are sending to us to solve these puzzles. — *RZ*

NEW PUZZLES

Puzzle 13 (Novices)

```
    V
    T
    R
  + E
   VS
 x VS
  VSS
```

Hints: T+T = E. Look at the samples and examples.

Puzzle 14 (Adepts)

```
     EC ATY CWU
X           TER
    WYL MYE RTE
    TUA YWE EC
  WURLT UYC T
  WRWEE WEW EWE
```

Hints: E and W in the top line when multiplied by R, E, and T of the second line, yield plenty of interlocking clues. However, if you get stuck, look on page 57 for some specific hints. But, try it first.

Puzzle 15 (Geniuses)

$$Y\ D^{E}\ H\ E^{L}\ S\ C\ U\ L^{E}$$
$$+\ B^{U}\ B\ B\ Y\ E^{E}\ U\ E^{L}\ D^{E}$$
$$C\ A\ E\ H\ Y\ C\ C\ U$$

Hints: No hints to geniuses. They compete on equal terms with computers.

Puzzle 16 (Computers)

```
  NMT EYL
x       A
Y NUR REY
```

Hints: Your computer won't need hints.

SAMPLES AND EXAMPLES

Puzzle 1 — *Puzzle 1 Solution*

```
 B     5
 B     5
 B     5
--    --
CB    15
```

The number 5 is the *only* digit that works.

Puzzle 2 — *Puzzle 2 Solution*

```
 R     9
 R     9
 T     1
--    --
TR    19
```

The number 9 is the *only* digit that works.

Puzzle 3 — *Puzzle 3 Solution*

```
  AC      32
 xAC     x32
----    ----
  FH      64
 JF      96
----    ----
YBCH    1024
```

The letter A can *only* have the value 3. If A were more than 3, three letters would be needed in place of JF; if A were less than 3, the total would not reach 1,000.

Puzzle 4 — *Puzzle 4 Solution*

```
  A, ABB, FCB      8,844,534
+ H, KCF, FKB     +1,235,524
-------------    -----------
HM, MAM, MFA      10,080,058
```

In column one, A *must* be an *even* digit. Why? The H must be a 1. Right? So HM is easy. That solves for B in column one; then column four; then column three . . . Get it? Stay with it a while and the logic starts jumping off the page.

COMMENTS FROM NINE HEX (JACK CREHORE)

Here are a few elementary hints for solving puzzles. Test these hints on each letter, first singly, then its relationship to other letters. Check to see if the letter: a) represents a 0, 1, 5 or 6; b) is a large or small digit; c) is an even or odd digit; d) generates a carry, when added to or multiplied by another digit.

The editors and I agree that we must not delay showing another solution device: the Elimination Table or "Is-Not" chart. Observe the 10 by 10 grid of squares:

"Is-Not" Chart

	0	1	2	3	4	5	6	7	8	9
A		x								
B		x								
C		x								
F		x								
H		x								
J		x								
K								K		
M	x	x	x	x	x					
N		x								
R	x	R	x	x	x	x	x		x	

The digits 0-9 are put along the top of the chart; the letters of the puzzle down the left edge. When you think you have a value for a letter—say, 7 for K—write K in the 7s column, in the square that is on the row labeled K. Then cross out every vacant square that is in the 7s column, and every vacant square on the K-line. That is, indicate K *is not* 1-6 or 8-9, and 7 *is not* any of the remaining letters.

Suppose you now find, as part of the overall problem:

```
 M
+M
RT
```

You know RT is at least 10, and therefore M is 5, 6, 8 or 9. (Remember: K is 7.) This fact means that M *cannot* be 0, 1, 2, 3 and 4. Cross out these last five squares on the M line. Based on what M can be, RT cannot be more than 18. Thus, R is 1. Write R in the 1s column of the Elimination Table, in the row labeled R, and cross out all other squares along the R line and 1s column.

See how it works? Good!

In the last issue, we printed the first five puzzler postcards we received. Here I acknowledge a few more, joyful at the happy chord you all strike—mostly, in refreshing *me* in arithmetic.

Dr. Dave: I appreciate your scholarly critique on the whole syndrome with which we are all having so much fun. Your second paragraph is *so* apt: "I was tempted to write a program for the solution of Puzzle 7, but I believe that the human ability to discover clues and adapt to new discoveries renders a human solution more efficacious and less troublesome than writing a program. . . . I enjoyed the challenge provided by Nine Hex. My compliments to him . . . "

Scott: Scholarly analyses! Your programming was so swift I heard you calling, "Look Ma—no hands." I sense an old timer.

Willz: Oh! You *are* an old timer! My earliest listing in the National Puzzlers' League directory was 1939. I've been too preoccupied since to rejoin. Now, I'm ready to get back in. I am sure *we* started the *fun* with puzzling, with our national conventions.

Prof. Jim Householder: From Humboldt State Univ., sends us a simple—uh . . . well, two-word example. He says there's a math fad on at HSU.

```
 TWO
+TWO
FOUR
```

— NINE HEX

PUZZLERS' SOLUTION TABLE

Name	9	10	11	12	Total Solutions
	Current Puzzle				
DAB	●	●	●	●	8
The Kid	●	●	●	●	8
Polarbaer	●	●		●	8
TWW					4
SPOCK					2
D. Hubbard					4
C.A. Moore					4
S. Schram					4
D. Marquis					4
B. Baum					4
R.H.(?)					1
M. Richter					1
C.I. Goldman					4
K8VDU					4
Crazy Man					4
S.R. McEntee					4
DADDIDWJM					4
J. Dushey					4
J. Kmoch					4
GEDASM					4
R. Washburne					4
Kettenhoffen					4
Cyrano					4
B. Potter					4
CTEIN					4
Hobo					4
Willz					4
K. Lewis					4

Legend: (●) By Hand
(X) By Computer

*Puzzle 11 from the last issue is solvable without setting D to zero! Puzzle 12 did have a typo. (See the introduction on the previous page.) Hubbard's puzzle is tricky and it has a solution. Look again! DAB discovered an error in the MAJOR * WOMAN puzzle. He says the line ICJIAO should be ICJMAO. DAB got DADDIDWJM'S name anyway. Oh, Yes! With the partial product change DAB says the puzzle solution line is DDJJIDWJM. As Polarbaer suggests, we will do a better job of "poorfreadin" in the future.* — *RZ*

Strange things seem to happen, says British writer Warwick Slough, when people and computers get together. Confronted by the temptations of a powerful machine, a man inevitably shows his true character. To that insight, add the fantasy of a Swiss vacation with a beautiful woman, and you have a classic Warwickian story. (Mr. Slough last appeared in these pages in the March-April issue.) —*LB*

INTERRUPT

BY WARWICK SLOUGH

Jeanette McPherson was an archetypal secretary of the 1980s. As the sun slanted through the venetian blinds of her office window, she peered at the green lettering flickering on a small VDU screen, while her fingers caressed the keys on the adjacent keyboard. She was using the small word processing system recently introduced by the insurance brokerage where she worked.

With a puzzled frown she scanned the screen's contents. The text was not what she expected. "Must have set up the wrong floppy disc," she mused. Still, the record appeared to have an interesting content. She flicked three keys, and the text moved forward on the screen. "Therefore in the present circumstances Dataret would seem to be a reliable investment for investors seeking steady returns. Security of computer-stored information will be an expanding field of activity, and Dataret has the essential expertise and staff to implement its systems." Although she did not recognise the record, she instructed the printer to make a hard copy, almost from habit before replacing the floppy disc with the one she required.

Later that morning she collected a heap of concertina-like folded paper and divided it into appropriate records. The strange record bore the title "Commercial Security Systems." She laid it at one end of her desk, to be dealt with when time allowed. At the lunch break time had still not allowed, and the record lay in a solitary state, to be discovered by the chief accountant, Martin Merton, when he arrived to take her to lunch. (A major part of Martin's extracurricular activities concerned Jeanette: wining, dining and even finding her.) As he digested the information from the printed out-put, the suspicion of a smile flitted over his face. The encryption algorithm used by Dataret was fully described, and he spent the next 20 minutes preparing a Xerox copy of the relevant pages.

Martin caught up with Jeanette at the "Jug and Bottle." She had reached the second gin and tonic condition (speaking without inhibition but not yet loquacious). The remainder of the lunch was unremarkable. No word was mentioned concerning security of data systems.

In the afternoon Joyce Brabingham approached Jeanette. Normally she was slightly condescending to the other staff (since she was a personal assistant to the managing director), but today her face was marked with a worried frown. "Jeanette, have you by any chance seen a floppy disc with a report concerning commercial security systems in the office?" she piped. "Sure," responded Jeanette, "I think it's over on the side table." With a grateful sigh, Joyce pounced on the intrusive disc.

Some eight weeks later Martin was making final arrangements to fly to Switzerland for the business trip of a lifetime. He was working with a VDU currently linked into a private Swiss computer network. He checked again the data which were already as familiar to him as his breakfast eggs.

Heavy Engineering and Maintenance Co.
Current Assets $2,000,000

Energy Transmission and Conversion Corp.
Current Assets $1,250,000

Semiconductor Technology Ltd.
Current Assets $2,650,000

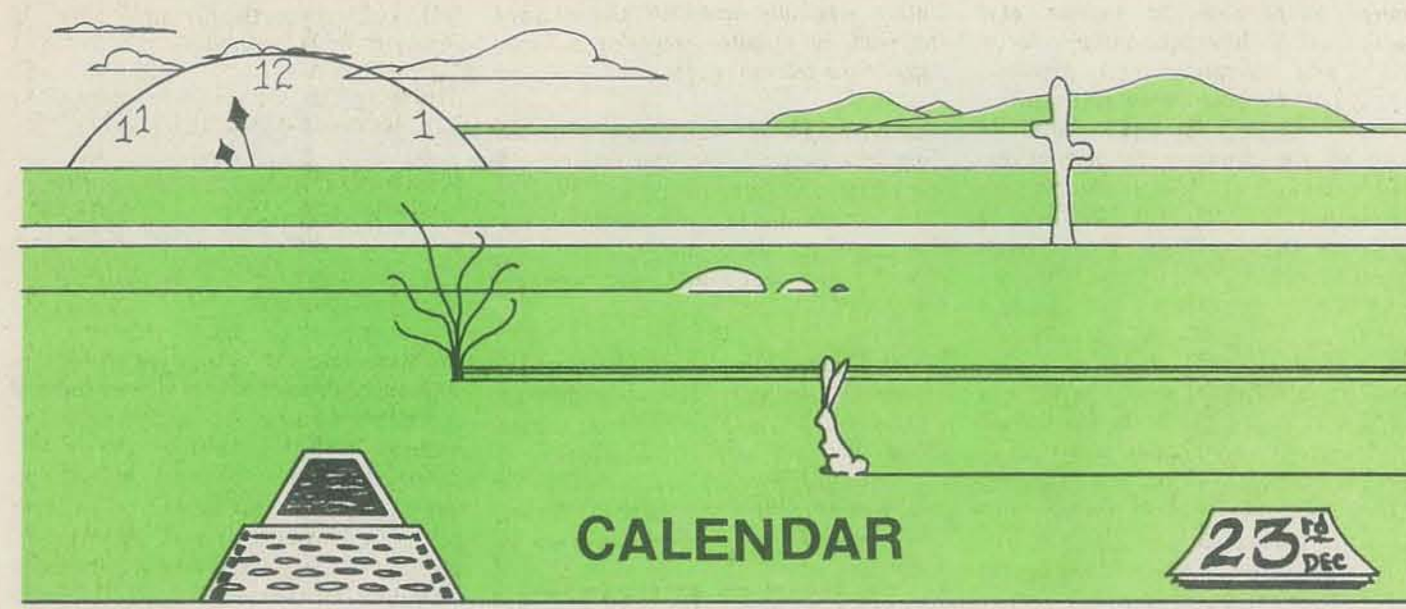
he read from the screen. No such firms existed, of course, but these were some of several apparently new customers whose business Martin's insurance company had taken on in the last few weeks. Furthermore, funds had been transferred to cover payments on a number of hypothetical claims, creating sizeable deposits of money in various Swiss accounts. With a satisfied nod he switched off the terminal. As he leaned back in his chair a glazed look came into his eyes and his mind conjured up visions. Il Dolce Vita (Swiss style) and Jeanette – yes, especially with Jeanette.

A tap at his office door ended his reverie and preceded the entrance of Wilkins, the company secretary. "Have you a few spare minutes, Martin?" he queried. "Got something that might amuse you."

Martin nodded wearily and indicated a nearby chair for his visitor. Wilkins was a hell of a bore but he'd better appear to give him some attention. Wilkins gushed on, impervious to the malicious desires permeating Martin's mind. "Well, about two months ago we were trying out a computer security system – Dataret I think they called themselves. Anyway, I was just chewing the fat with our computer people, and it seems there's some snag with the implementation of the Dataret system. Meanwhile, of course, all existing information about our customers has been coded up using their methods.

"Now they've got to change the lot to some new firm's system we're adopting. Old Briggs is mad as hell – says how can he be expected to keep staff when they have to do everything twice? Anyway thought it might amuse you," he chortled.

"By the way you're not looking too good – probably been overworking. Why don't you take a few days off. I believe Switzerland's pretty good at this time of year."



CALENDAR

BY H. FRANK ANDERSEN, M. D.

For anyone who has a busy calendar and is constantly having to juggle dates, determine differences between dates or add and subtract with dates, here's the program to let your computer do it for you. The author is an obstetrics and gynecology resident at the University of Michigan, and developed the program out of a study of the projection of delivery dates in pregnant patients. — *TD*

Microcomputer applications in the home, business, or laboratory frequently involve entering dates. Occasionally, part of the program may involve manipulating the dates, for example, determining the number of days between two dates or adding or subtracting a number of days from a given date. If the application involves only short time periods, this can be done by a relatively simple loop which essentially counts one day at a time. This kind of process is used by some programs available commercially; however, such a program can be slow for long time periods.

I recently came across this problem in a situation where I wanted to project a final or termination date from information on several earlier dates. What I needed was a system which gave:

1. A single number with a constant format which uniquely represented a single date.
2. Routines to determine the difference between two dates or to add a number of days to a date and determine the new date.
3. An approximate date that could be recognized from this number, and could be used for preliminary calculations on a hand calculator.

YR.DAY

The format I decided upon uses a real number in which the integer portion is the year and the decimal portion is the elapsed number of days of the year on the given date: YY.DDD, or if the century is important, YYYY.DDD. I call this the "YR.DAY" format. The decimal portion is a modulo 365 (or modulo 366 in a leap year) number divided by 1000. Thus, Jan 1, 1978 is 78.001 and Dec 31, 1978 is 78.365. Dec 31, 1976 would be 76.366 since 1976 is a leap year. It would be necessary to represent this as 1976.366 if dates will occur in different centuries within the program.

The subroutine beginning at line 1000 converts a date (M,D,Y) into YR.DAY format (D1) by adding the days in each preceding month, plus the date of the current month, then dividing by 1000, and adding the year. Using a reverse process, the subroutine at line 1300 converts YR.DAY (D1) back to a date (M,D,Y). The routines use loops with "GOTO" rather than "FOR-NEXT" statements because I discovered that problems result when I branch from within a "FOR-NEXT" loop with "I" as index to a subroutine also using a "FOR-NEXT" loop with "I" as index.

ADDITION-SUBTRACTION

The addition subroutine, line 1600, adds a specific number of days to a date in YR.DAY format by adding one year, or 365 or 366 days, at a time. The subtraction-two subroutine at line 2200 performs a similar subtraction process.

The subtraction-one subroutine at line 1900 determines the difference between two dates. Note that D1 must be a later date, meaning a larger number in YR.DAY format than D2. The routine subtracts the .DAY portion with an appropriate "carry" of 365 or 366 days if necessary, and the YR. portion, then multiplies the YR. difference by 365, and adds an appropriate number of days for leap years. Line 2090 increments one day each time a leap year is crossed and adds that to the result. Lines 2100 and 2110 modify that for centuries.

```
800 REM            ***** CALENDAR *****
810 REM
820 REM           H. FRANK ANDERSEN, M.D.
830 REM               117 LØNGMAN LANE
840 REM          ANN ARBØR, MICHIGAN  48103
850 REM
860 REM    IF ACCURACY ACRØSS CENTURIES IS NØT DESIRED, DELETE LINES
870 REM  INDICATED WITH  "\ REM <<"   (2 LINES IN EACH SUBRØUTINE).
880 REM
890 REM
1000 REM  SUBRØUTINE TØ CØNVERT DATE TØ YR.DAY      **********
1010 REM   **  INPUT - M (MØNTH)
1020 REM   **          D (DAY)
1030 REM   **          Y (YEAR)
1040 REM   **  ØUTPUT - D1 (YR.DAY)
1050 REM   **           M
1060 REM   **           D
1070 REM   **           Y
1080 REM   **  VARIABLES USED\ M, M1, M2, D, D1, D9, Y
1090 REM
1100         DATA 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31
1110         RESTØRE
1120         D9 = 0
1130         M2 = 1
1140 REM
1150      IF M2 = M THEN 1210
1160         READ M1
1170         D9 = D9 + M1
1180         M2 = M2 + 1
1190       GØTØ 1150
1200 REM
1210      IF M2 < 3 THEN 1250
1220         IF Y/400 = INT(Y/400) THEN 1240 \ REM <<
1230         IF Y/100 = INT(Y/100) THEN 1250 \ REM <<
1240         IF Y/4   = INT(Y/4)   THEN D9 = D9 + 1
1250      D1 = (D9 + D)/1000 + Y
1260 REM
1270      RETURN
1280 REM
1290 REM
1300 REM  SUBRØUTINE TØ CØNVERT YR.DAY TØ DATE      **********
1310 REM   **  INPUT - D1 (YR.DAY)
1320 REM   **  ØUTPUT - M (MØNTH)
1330 REM   **          D (DAY)
1340 REM   **          Y (YEAR)
1350 REM   **          D1
1360 REM   **  VARIABLES USED\ D, D1, M, M1, Y
1370 REM
1380 REM
1390            RESTØRE
1400            Y = INT(D1)
1410            D = 1000*(D1 - Y)
1420            M = 0
1430 REM
1440         M = M + 1
1450         READ M1
1460 REM
1470      IF M <> 2 THEN 1520
1480         IF Y/400 = INT(Y/400) THEN 1500 \ REM <<
1490         IF Y/100 = INT(Y/100) THEN 1520 \ REM <<
1500         IF Y/4   = INT(Y/4)   THEN M1 = M1 + 1
1510 REM
1520      IF D > M1 THEN D = D - M1 ELSE 1550
1530         GØTØ 1440
1540 REM
1550      RETURN
1560 REM
1570 REM
1600 REM  ADDITIØN SUBRØUTINE                       **********
1610 REM   **  INPUT - D1 (YR.DAY)
1620 REM   **          D2 (NØ. ØF DAYS)
1630 REM   **  ØUTPUT - D3 (YR.DAY WHERE D3 = D1 + D2 )
1640 REM   **          D1
1650 REM   **          D2
1660 REM   **  VARIABLES\ D0, D1, D2, D3, D9
1670 REM
1680 REM
1690          D3 = D1
1700          D9 = D2
1710 REM
1720      IF INT(D3)/4   = INT( INT(D3)/4)   THEN D0 = 366 ELSE D0 = 365
1730      IF INT(D3)/100 = INT( INT(D3)/100) THEN D0 = 365 \ REM <<
1740      IF INT(D3)/400 = INT( INT(D3)/400) THEN D0 = 366 \ REM <<
1750 REM
1760      IF D9 >= D0 THEN D3 = D3 + 1 ELSE 1800
1770          D9 = D9 - D0
1780          GØTØ 1720
1790 REM
1800      D3 = D3 + D9/1000
1810          IF D3 - INT(D3) > D0/1000 THEN D3 = D3 + 1 - D0/1000
1820 REM
1830      RETURN
1840 REM
1850 REM
1900 REM  SUBTRACTIØN - ØNE SUBRØUTINE              **********
1910 REM   **  INPUT - D1 (YR.DAY)
1920 REM   **          D2 (YR.DAY)
1930 REM   **  ØUTPUT - D3 (NØ. ØF DAYS WHERE D3 = D1 - D2 )
1940 REM   **          D1
1950 REM   **          D2
1960 REM   **  VARIABLES\ D1, D2, D3
1970 REM
1980 REM
1990          D3 = 0
2000 REM
2010      IF D1 >= D2 THEN 2050
2020          PRINT "SIZE ERRØR - D1<D2"
2030          GØTØ 2130
2040 REM
2050      D3 = D1 - D2
2060          IF D3 - INT(D3) > .365 THEN D3 = D3 - .635
2070      D3 = (D3 - INT(D3))*1000 + INT(D3)*365
2080 REM
2090      D3 = D3 + INT(INT(D1)/4 + .8)    - INT(INT(D2)/4 + .8)
2100      D3 = D3 - INT(INT(D1)/100+.995) + INT(INT(D2)/100+.995)\REM<<
2110      D3 = D3 + INT(INT(D1)/400+.998) - INT(INT(D2)/400+.998)\REM<<
2120 REM
2130      RETURN
2140 REM
2150 REM
2200 REM  SUBTRACTIØN - TWØ SUBRØUTINE              **********
2210 REM   **  INPUT - D1 (YR.DAY)
2220 REM   **          D2 (NØ. ØF DAYS)
2230 REM   **  ØUTPUT - D3 (YR.DAY WHERE D3 = D1 - D2 )
2240 REM   **          D1
2250 REM   **          D2
2260 REM   **  VARIABLES\ D0, D1, D2, D3, D9
2270 REM
2280 REM
2290          D3 = D1
2300          D9 = D2
2310 REM
2320      IF INT(D3-1)/4 = INT( INT(D3-1)/4) THEN D0 = 366 ELSE D0 = 365
2330      IF INT(D3-1)/100 = INT( INT(D3-1)/100) THEN D0 = 365 \ REM <<
2340      IF INT(D3-1)/400 = INT( INT(D3-1)/400) THEN D0 = 366 \ REM <<
2350 REM
2360      IF D9 > D0 THEN D3 = D3 - 1 ELSE 2400
2370          D9 = D9 - D0
2380          GØTØ 2320
2390 REM
2400      IF D9/1000 >= D3 - INT(D3) THEN D3 = D3 - 1 + D0/1000
2410          D3 = D3 - D9/1000
2420 REM
2430      RETURN
```

Each subroutine contains a list of input and output variables, and a list of the variables used within the routine. As printed, the program requires about 4000 bytes; however, if the REM statements are removed the memory requirement drops to 2100 bytes. If accuracy across centuries is unnecessary, and statements are compacted, the whole package can probably be put in under 1500 bytes. Of course, only those routines necessary for a specific application need be used.

POTENTIAL "GOTCHA'S"

The main problem with any dating routine is leap years. A century year is not a leap year unless it is divisable by 400, though. This system, known as the Gregorian calendar, was instituted in 1582 A.D. when 10 days were simply dropped from the year to coordinate the previous system, the Julian calendar, with astronomical measurements. Britain and the American Colonies, however, did not begin to use the Gregorian calendar until 1752. Thus, if for any reason calculations are done involving dates near these years, inaccuracies may result. Also, the year 4000 will not be a leap year. This is not currently included in the program, and an error of one day will result in the year 4000 A.D. The program is accurate across leap years and centuries except for the problems mentioned. If accuracy across centuries is not needed, the lines indicated by \REM<< can be deleted.

This program was written in Northstar BASIC, Version 6. It has also been run in CP/M BASIC E; however, I found that occasionally it would produce a YR.DAY with more than three decimal places (e.g., 78.364997). This can be easily corrected by adding a line to round to the nearest 0.001 at the end of each routine.

In writing this article I found the following references to be of value:

1. R. W. Bemer, "Time," *Interface Age,* Vol. 4, p. 74, Feb. 1979.

2. J. D. Robertson, "Remark on Algorithm 398," *Commun. ACM,* Vol. 15, p. 918, Oct. 1972.

3. R. A. Stone, "Tableless Date Conversion," Algorithm 398, *Commun. ACM,* Vol. 13, p. 621, Oct. 1970.

TRS-80: Elementary Math

BY KAREN D. CHEPKO

The Chepko software writing team is expanding! Here is Karen's first program. Milan Chepko, Karen's husband, writes a lot of software for RC. He may have to get a second computer soon, as Karen discovers the "Joys of Computing."

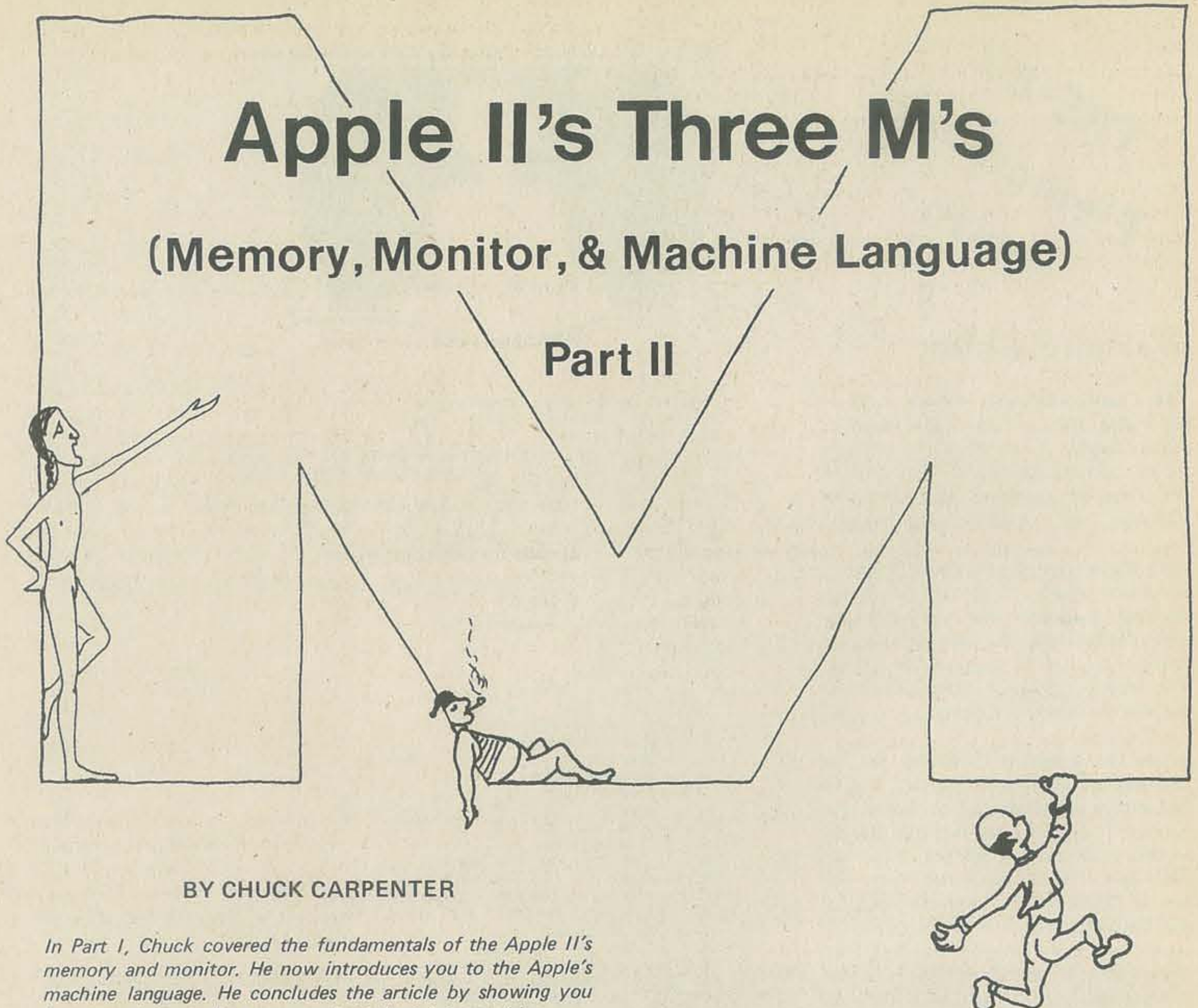
Both Karen and Milan are not "trained" computer people. What they are doing, building programs and experimenting with a home computer, presages what the micro-revolution is going to be about. You may not know it, Karen and Milan, but you are pioneers. Keep at it . . . .– RZ

After two years of listening to my husband talk about his computer, I felt I had absorbed enough to write some programs of my own. Besides, why should all that equipment sit idle during the day while he's at work? Since our 7-year-old son is interested in both mathematics and the computer, I decided to write an elementary math program for him. The result was several hundred seemingly unorganized lines that produced addition and subtraction. Milan patiently gave me many helpful hints over the next few weeks and I was able to shorten the program while including multiplication and division.

The program presents a different set of problems for kids in grades K through 1, 2, and 3 and above. In setting up the problems, I used "print using" statements. This kept the right-hand margins even, allowing the problems to be shown in the conventional manner.

The testing section compared the input answer (S) with the correct answer (J). Up to four attempts to answer are allowed before the correct answer is displayed. I did not use score keeping routines because this program was to be for fun and not for pressure.

```
10 REM "ELEMENTARY MATH BY KAREN CHEPKO"
20 REM "THIEF RIVER FALLS, MINNESOTA 56701"
30 CLS : DEFINT A-Z
40 INPUT "HELLO! WHAT IS YOUR NAME" ; N$ : PRINT
50 INPUT "WHAT GRADE ARE YOU IN (KINDERGARTEN AND BELOW = 0)";G
60 REM "INSTRUCTIONS FOR DIVISION"
70 CLS : PRINT "DIVISION PROBLEMS WILL BE DISPLAYED AS FOLLOWS:":PRINT
80 PRINT "  16" : PRINT "------ = ?" : PRINT "   5" : PRINT
90 PRINT "THE QUOTIENT IS ? (THE ANSWER IS 3)"
100 PRINT "WITH A REMAINDER OF ? (THE ANSWER IS 1)"
110 FOR T=1TO1500 : NEXT T
120 PRINT : PRINT : PRINT "HAVE A PENCIL AND PAPER HANDY, ";N$;". LET'S GET STARTED!!"
130 FOR T=1TO2000 : NEXT T
140 REM "DETERMINES INITIAL VARIABLES"
150 IF G<2 , X=10
160 IF G=2 , X=100
170 IF G>2 , X=10000
180 CLS : V=RND(2) : W=RND(2)
190 S=0 : L=0 :Y=RND(X) : Z=RND(X)
200 IF V=1 , C=RND(X) ELSE C=0
210 REM "DETERMINES FUNCTIONS"
220 IF Y<Z AND W=1 , D=1 : A=Z : B=Y : J=A+B+C
230 IF Y<Z AND W=2 , D=2
240 IF D=2 AND G<=2 , A=Y : B=Z : J=A*B
250 IF D=2 AND G>2 , A=RND(200) : B=RND(150) : J=A*B
260 IF Y=>Z AND W=1 , D=3 : A=Y : B=Z : J=A-B
270 IF Y=>Z AND W=2 , D=4
280 IF D=4 AND G<2 , A=Y : B=Z : J=A/B
290 IF D=4 AND G=2 , A=Y : B=RND(10) : J=A/B
300 IF D=4 AND G>2 , A=RND(3200) : B=RND(100) : J=A/B
310 IF D=4 AND A<B , GOTO180
320 REM "SETS UP PROBLEMS"
330 IF D=1 , Q$="SUM" : P$="+#####"
340 IF D=2 , Q$="PRODUCT" : P$="X#####"
350 IF D=3 , Q$="REMAINDER" : P$="-#####"
360 IF D=4 , Q$="QUOTIENT" : P$="######"
370 U$="######"
380 IF D=1 AND C>0 , PRINT USING U$;C
390 IF D=2 AND A<B , PRINT USING U$;B ELSE PRINT USING U$;A
400 IF D=4 , PRINT "------ = ?"
410 IF D=2 AND A<B , PRINT USING P$;A ELSE PRINT USING P$;B
420 IF D<>4 , PRINT "------" : PRINT
430 PRINT "THE ";Q$;" IS"
440 INPUT "  ";S
450 IF D=4 , INPUT " WITH A REMAINDER OF";L
460 REM "TESTS ANSWERS"
470 IF D=4 , K=B*J : R=A-K
480 IF D=4 AND L<>R , GOTO520
490 IF J<>S GOTO520
500 PRINT : PRINT "VERY GOOD, ";N$;"! YOU GOT IT RIGHT ON THE FIRST TRY! "
510 FOR T=1TO2500 : NEXT T : GOTO180
520 FOR M=1TO3
530 INPUT "OOPS! TRY AGAIN. THE ANSWER IS";S
540 IF D=4 , INPUT " WITH A REMAINDER OF";L ELSE PRINT
550 IF D=4 AND L<>R , GOTO 590
560 IF J<>S , GOTO590
570 PRINT : PRINT "MUCH BETTER, ";N$;"!!"
580 FOR T=1TO2500 : NEXT T : GOTO180
590 NEXT M
600 PRINT "THE ";Q$;" WAS";J
610 IF D=4 , PRINT " WITH A REMAINDER OF";R
620 FOR T=1TO2500 : NEXT T
630 PRINT "LET'S TRY ANOTHER ONE, ";N$;"!"
640 FOR T=1TO1500 : NEXT T : GOTO180
```



Apple II's Three M's

(Memory, Monitor, & Machine Language)

Part II

BY CHUCK CARPENTER

In Part I, Chuck covered the fundamentals of the Apple II's memory and monitor. He now introduces you to the Apple's machine language. He concludes the article by showing you how to write a small program and use the mini-assembler.

If you are a beginning Apple user, read on. You will begin to discover how to really use your machine. — *RZ*

There are two ways to put a machine language program in memory. One way is directly in HEX codes, using monitor commands. The other way uses the built-in mini-assembler utility. However, before we try to write an assembly language program, some knowledge is needed of the machine parts and the assembly language.

Instruction Set

Microprocessors use a set of codes for commands and instructions. The 6502 microprocessor has an instruction set that has 55 three letter codes. The codes are called *mnemonics* (*ni-mon-icks*—meaning easy to remember). By themselves, the mnemonic instruction codes can't always tell the 6502 what to do. Additional information called an *operand* is used with most instruction codes. Here are some examples:

Mnemonic	Operand
LDA	#$B5
JSR	$FDED
RTS	

Operation Codes

Each mnemonic instruction code also has a corresponding HEX value called an operation code. These opcodes are recognized by the system monitor and coverted to binary values for the 6502. Actually, the computer only recognizes binary numbers, but using HEX numbers to respresent binary numbers is a step toward making programming easier. Binary to HEX conversion is a task handled by the Apple II monitor.

Writing programs in hexadecimal opcodes and data can still be awkward. The assembly language, with mnemonic instruction codes and operands, is another step toward simplification. However, HEX opcodes remain a significant and necessary part of 6502 machine language programming.

Address Modes

The operands with each instruction code can identify which opcode to use for the instruction. Operands also tell the computer which address mode to use. Address modes instruct the computer to do something specific with the contents of the operand. There are several possible address modes that can be used with many of the instructions. Depending on the results and type of program, different address modes can be used with the instruction code. Later we will describe just three address modes. More information is available from the references at the end of the article.

Some Words on the 6502

The 6502 microprocessors have internal read/write (RAM) memory called registers. These registers allow the programmer to move instructions and data into and out of the microprocessor.

One register is called the accumulator (or A register), and two others are the X and Y registers. The accumulator is the most important register in the 6502. Many program steps will put data in the accumulator then put it into some memory location. Sometimes an operation is performed directly on the value in the accumulator. *Load* and *store* are two such operations. Load causes a value to be placed in a 6502 register. Store takes data from a 6502 register and puts it into an external memory location.

Mnemonics, Addressing and Opcodes

Instruction codes used for the load and store accumulator operations are LDA and STA. The three letter mnemonic is taken from characters in the instruction.

```
LDA   (LoaD Accumulator)
STA   (STore Accumulator)
```

The instruction LDA does one of two things: 1) loads the value in the operand into the accumulator or, 2) loads the value found at the address in the operand into the accumulator. The 6502 knows what to do by the way you write the operand. Depending on the form used, the addressing mode is defined accordingly. Here are two examples for the LDA instruction.

```
1.  LDA  #$B5    (Immediate addressing mode)
2.  LDA  $0300   (Absolute addressing mode)
```

In example 1, the # sign indicates that the accumulator is to be loaded with $B5. ($ in front of a number means HEX.) Example 2 tells the computer to get the value found at location $0300, and load it into the accumulator.

Instruction STA tells the 6502 to take the current value in the accumulator and store it in the address specified by the operand. For example, STA $0300 means take the value in the accumulator and store it in location $0300. Two other instructions we will use are JSR (Jump to SubRoutine) & RTS (ReTurn from Subroutine).

We said that mnemonic instruction codes could be represented by HEX opcodes. Here is a list of the codes used so far:

Op-code	Instruction	Address Mode
$A9	LDA	Immediate
$AD	LDA	Absolute
$80	STA	Absolute
$20	JSR	Absolute
$60	RTS	Implied

There are many other instruction codes, addressing modes and opcodes in the 6502 instruction set. A summary of these is shown on pages 100 to 105 of the *Apple II Reference Manual.*

Assembled Code Conventions

To construct an assembled program, you need to know something about the conventions for an assembled single byte, two byte and three byte instruction. Also, you must know the arrangement of the bytes in the three byte instruction. Depending on the instruction used and the data in the operand field, the opcode is assembled followed by the required number of data bytes. Immediate mode addressing uses the opcode then one byte of data. Absolute mode uses the opcode and two bytes of data. When assembled, the least significant operand byte is first, then the most significant byte.

```
$C030 ◄──HEX Address              │
  │└──────Least Significant Byte  │  Reversed when
  └───────Most Significant Byte   │  assembled.
```

This convention was designed into the 6502 to provide for special page 0 address modes.

Implied mode (the RTS instruction for example) uses only a single byte opcode. The instruction itself includes all the information needed for the desired result. Instruction RTS is used when you call one program from another. The return from subroutine returns you back to a point where you want to continue another program (or subroutine).

Putting It All Together

Now, let's build an example program and assemble it by hand. The program must 1) load the accumulator (LDA) with a value, 2) jump to a subroutine (JSR) that prints the contents of the accumulator on the screen, 3) jump to another subroutine (JSR) to 'beep' the bell, and 4) end the program (RTS).

First, write the assembly language program that will do these things. I'll provide you with the value for the accumulator and the subroutine addresses for the operands to get things going.

Inst.	Operand	Comment
LDA	#$B5	; load the accumulator with $B5
JSR	$FDED	; jump to character-out routine
JSR	$FBE4	; jump to bell routine
RTS		; make a definite return

Taking the above program, begin the hand assembly, have the program start at address $0300 and use consecutive addresses. Begin with the opcode for LDA immediate. The next memory location will contain the data in the operand. An opcode always has to be the first byte of data in your program. Most of the time it will be some form of the LDA instruction.

```
0300-  A9
0301-  B5
```

Look up the opcodes for each of the other mnemonic instruction codes and write them down. (We did this earlier so refer to that table.) Write the opcode for JSR in the next consecutive memory location, followed by the data in the operand. Remember the reverse sequence of the bytes of data in the operand.

```
0302-  20
0303-  ED
0304-  FD
```

Do the same thing with the next JSR and operand.

```
0305-  20
0306-  E4
0307-  FB
```

Complete the program with the single byte RTS instruction.

```
0308-  60
```

Here's how this program looks in the Apple mini-assembler format. Note that the opcode follows each address and is followed by the data defined by the operand.

Loct.	Mach. Code	Inst.	Operand
0300-	A9 B5	LDA	#$B5
0302-	20 ED FD	JSR	$FDED
0305-	20 E4 FB	JSR	$FBE4
0308-	60	RTS	

Apple II Mini-Assembler

Now let's try the mini-assembler to write a program. First, turn on your Apple and press the RESET key. You should see the asterisk (*) prompt and flashing cursor block in the lower left corner. In the following sequence, you will be typing in the underlined characters. The computer response is not underlined. Also, it is not necessary to use the $ character or leading zeros. The mini-assembler takes care of these things. The character ƀ (slash b) means type a blank with the space bar.

Step	Action	Comments
1.	F666G (RETURN)	; enter mini-assembler at $F666
2.		; prompt and cursor
3.	300:LDA #B5 (RETURN)	; first line to be assembled
4.	0300- A9 B5 LDA #$B5	; assembled output
5.	ƀ JSR FDED (RETURN)	; next line for assembly
6.	0302- 20 ED FD JSR $FDED	; assembled output
7.	ƀ JSR FBE4 (RETURN)	; continue as above
8.	0305- 20 E4 FB JSR $FBE4	; assembled output
9.	ƀ RTS (RETURN)	; last line to assemble
10.	0308- 60 RTS	; final assembled output

You have now assembled a program starting at location $0300. Now run the program using the following sequence:

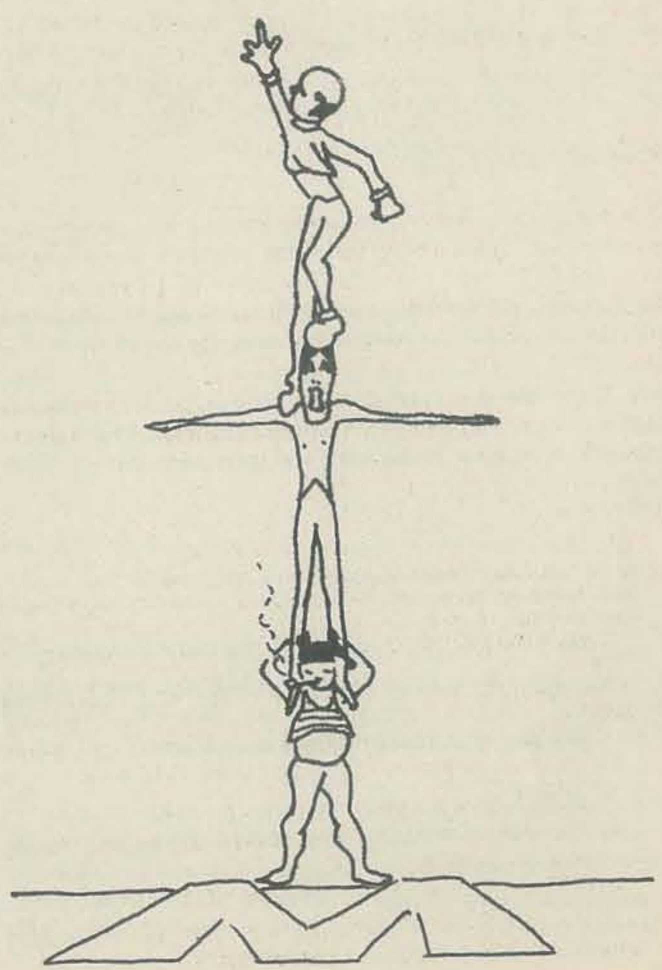
```
$300G                ; run program from assembler
5                    ; see a 5, hear a beep
```

What we just did was to run the program from inside the mini-assembler. This is what happened:

First, the start location in memory was indicated, address $0300. The G following $300 is the command to run the program. Inside the mini-assembler, the $ tells the computer to expect a monitor command.

Second, the value $B5 was stored in the accumulator. The immediate mode LDA command did this operation. The HEX value $B5 represents the decimal number 5.

The operands in the JSR instructions represent the locations of two subroutines in the Apple II monitor. A character output routine is at $FDED. The program calls on this routine



and the accumulator value is put on the screen. Our first instruction loaded the HEX value for a 5 into the accumulator. The routine at address $FBE4 is called BELL2. The program then calls on this routine to generate the 'beep' heard in the speaker. The program ends with the RTS instruction.

To exit from the mini-assembler, press RESET. An asterisk prompt will return. Now type 300L and press RETURN. A listing of 20 disassembled lines will appear on the screen. Only the first 5 lines include the character output and bell ringing routine. The other code shown is not a valid part of the program.

Try One By Yourself

You can run this example program as often as you want by typing 300G and RETURN. Try experimenting with different values in the accumulator. Numbers 0 to 9 are values $B0 to $B9. Letters A to Z are values $C1 to $DA. A space is $A0 and a carriage return is $8D. Write a program to print your name or the current date. Hint: use LDA immediate for each character you want to print. End a line with a carriage return, and end the program with RTS. Pages 68 to 71 in the *Apple II Reference Manual* list the monitor commands. Explore these and experiment. You can't do anything more than mess-up your own programs!

Assembling long programs will begin to become awkward with the mini-assembler. If you want to get serious with assembly and machine language, get an assembler such as the S-C Assembler II (see references). A good assembler allows you to insert and delete lines, label and comment program lines, perform text editing tasks, and it makes machine language programming nearly as easy as programming in BASIC.

What Did He Say?

This article has covered a lot of ground: Apple's memory and monitor were highlighted, the inner workings of these areas were briefly reviewed and the concepts of binary numbers and hexadecimal numbers were covered. We ended with a brief introduction to writing an assembly language program.

Still to be explored are symbolic operands, loop indexing, page zero features, and other monitor routines. That leaves a lot more to be covered, but we'll end it here for now.

References

1. *Apple II Reference Manual*, 1978, Apple Computer Co.
2. *The Newsletter for Apple II Owners*, Vol. 1 issue 2, pages 19-21, Rainbow, Box 43, Audubon, PA 19407
3. *S-C Assembler II*, SC Software, P.O. Box 5537, Richardson, TX 75080
4. *Aresco-Assembler/Text Editor*, Aresco, Box 43, Audubon, PA 19407
5. *6502 Programming Manual*, Several sources—Rockwell International, Synertec, Commodore (from the makers of the 6502 microprocessor)
6. *Programming the 6502*, Rodnay Zacks, Sybex, 1979
7. *How to Program Microcomputers*, William Barden Jr., Howard W. Sams & Co., 1978
8. *Programming a Microcomputer-6502*, Caxton C. Foster, Addison-Wesley Publishing Co., 1978
9. *Micro, The Magazine of the Apple, KIM, PET and Other 6502 Systems*, P.O. Box 3, So. Chelmsford, MA 01824

THE FURTHER ADVENTURES OF

FORTRAN MAN

Volume III
Episode 9

BY LEE SCHNEIDER & TODD VOROS

In our last episode, with less than a half-cycle remaining before the main body of the Resistance forces under General Wirewound begin their assault on Capital City, the other half of the Resistance embarks on a perilous journey across the data fields and over the Great Voltage Divide to reach the stronghold of the oldest and greatest of the Clans – Castle McIntel!

Linea, their commander, knows that without extraordinary help they will never be able to connect their decades with the General's in time to turn the flow of the battle. She also knows that such help can be obtained from none other than Fortran Man!

Our Hero, however, is unable to use most of his powers. When parts of his memory were inadvertently erased, much of his power went as well!

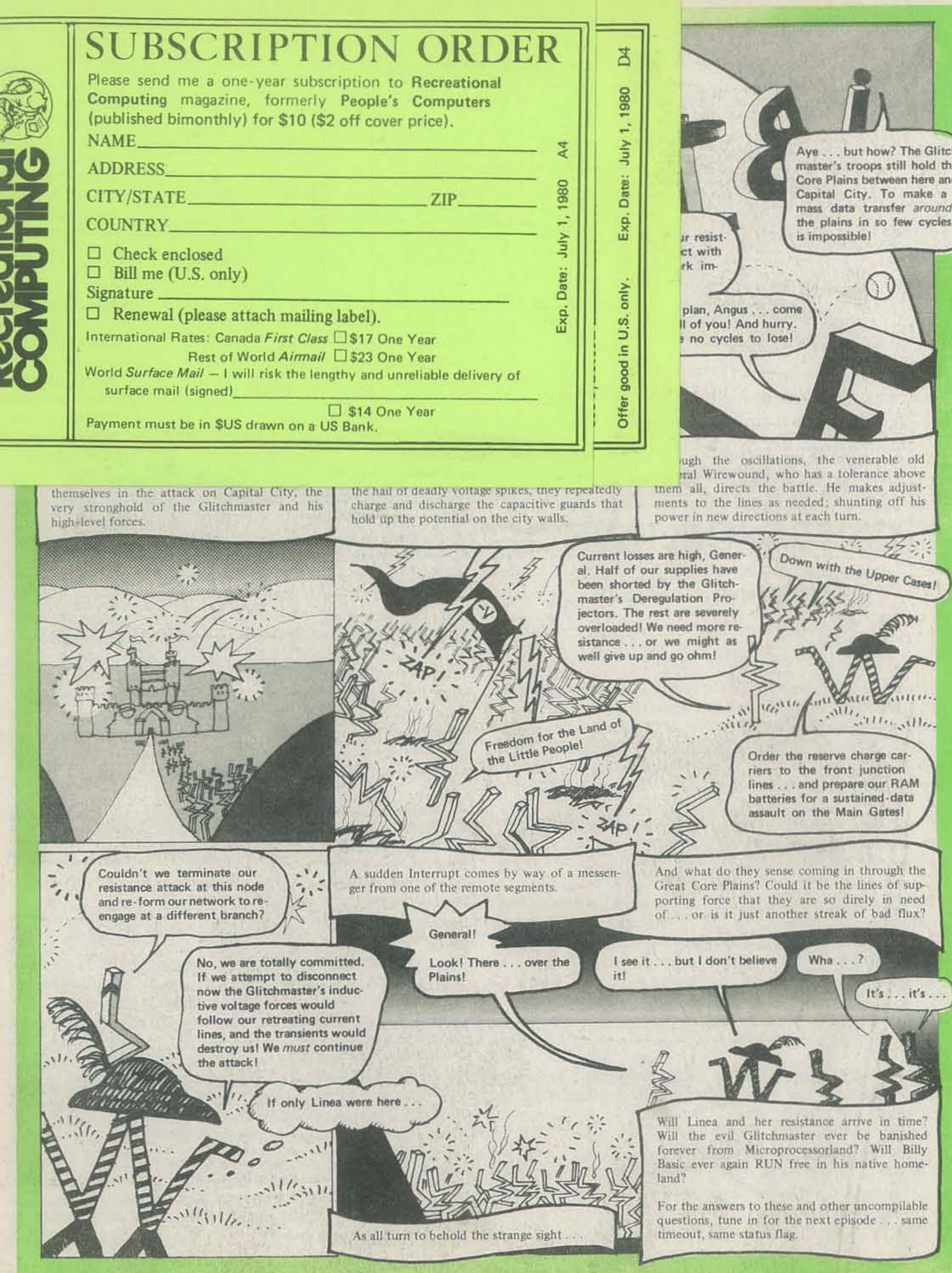
The solution? A memory-to-memory transfer with Billy Basic serving as the source file for information!

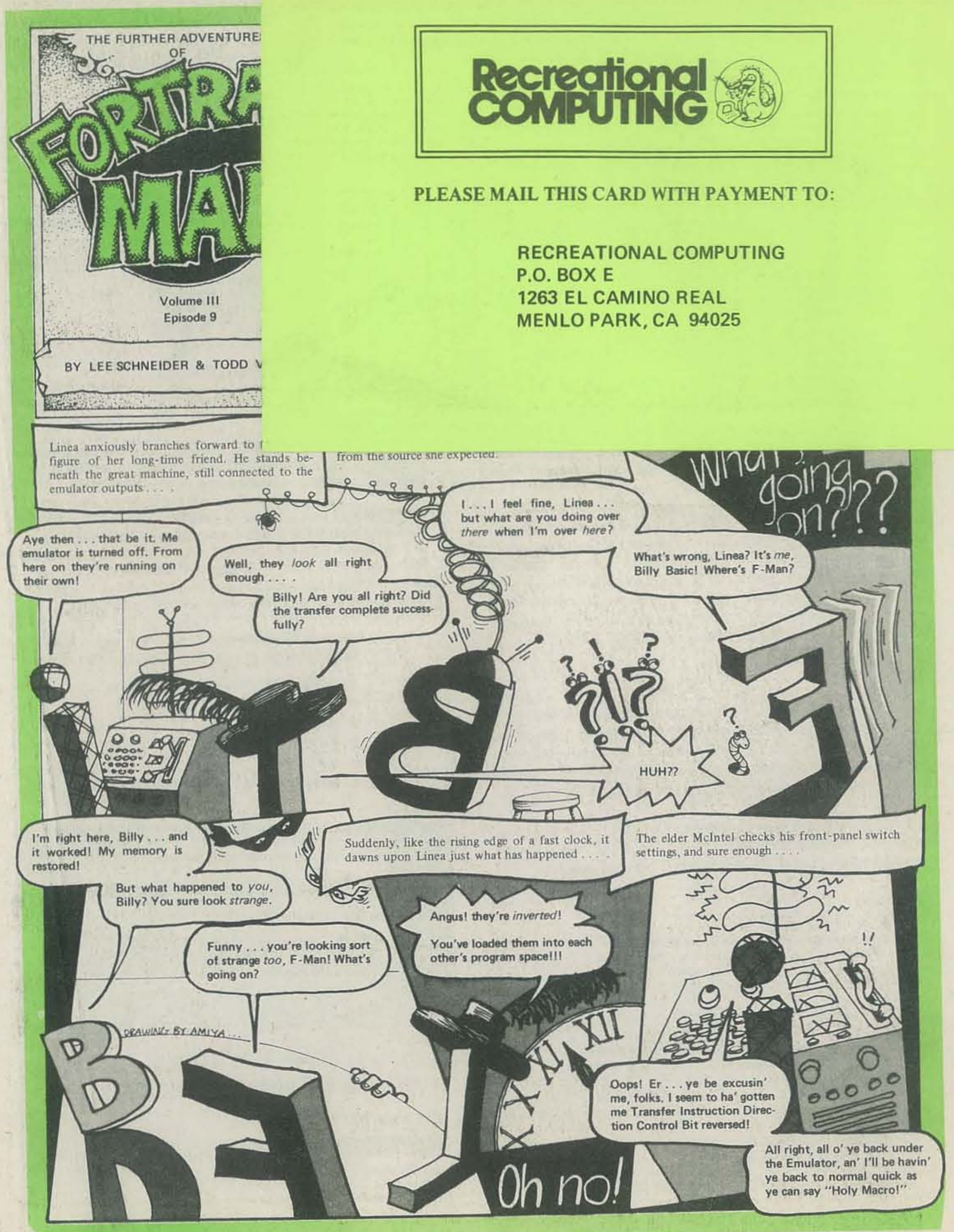
But this transfer requires special equipment . . . and thus the reason for their journey. For deep within the lower levels of Castle McIntel, below the huge Syntax Separators and Mneumonic Assemblers, lies the only remaining operational system of its kind in the land – an In-Circuit Emulator!

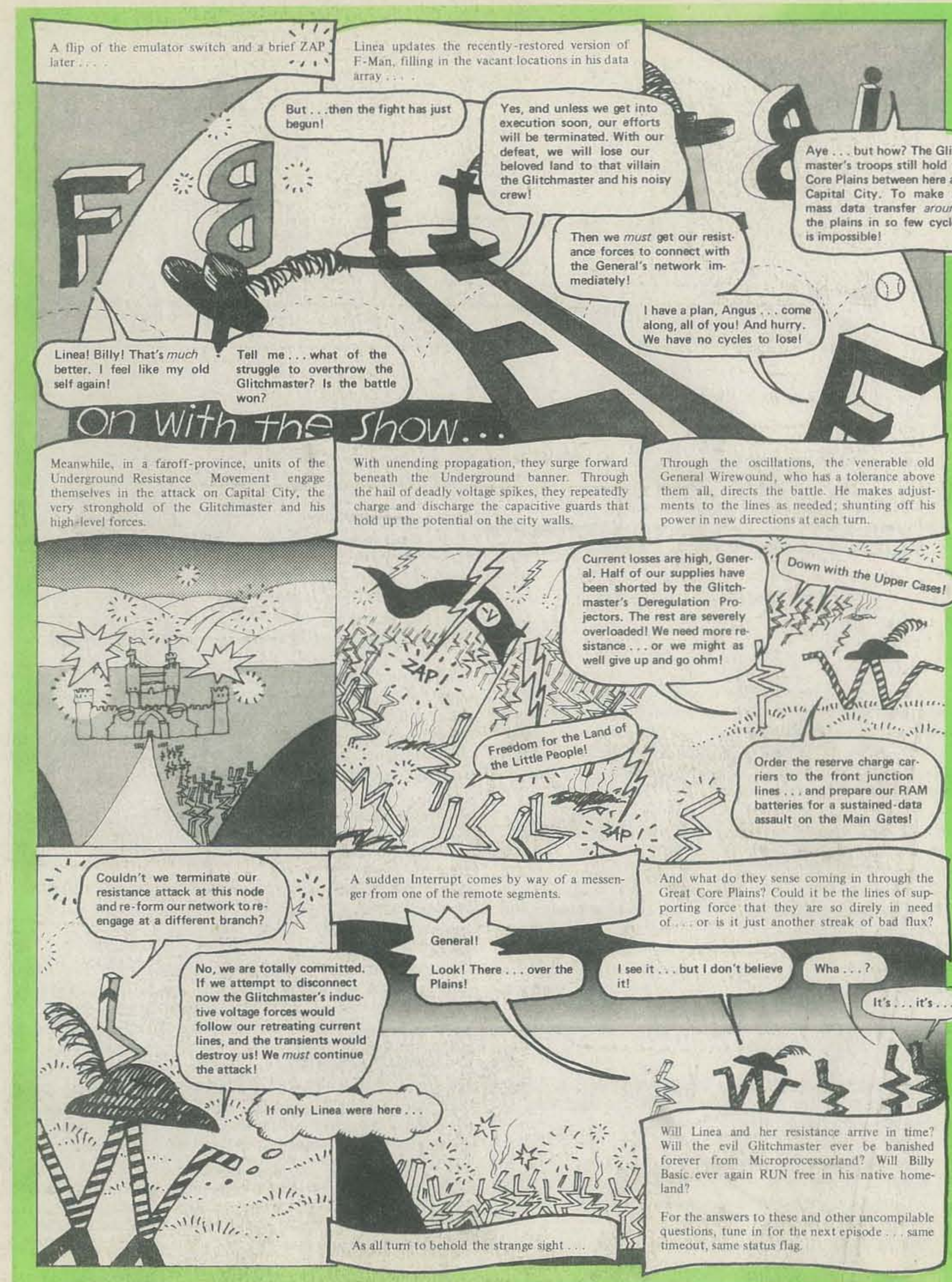
In grateful thanks for the return of their ancestral mascot, the Lockout Monster, the Clan agrees to admit them to the stronghold and allows them to utilize the rare and powerful device. Angus McIntel, leader of the Clan, leads them into the lower levels of the hardware, and there amongst layers of fine digital dust the emulator stands waiting.

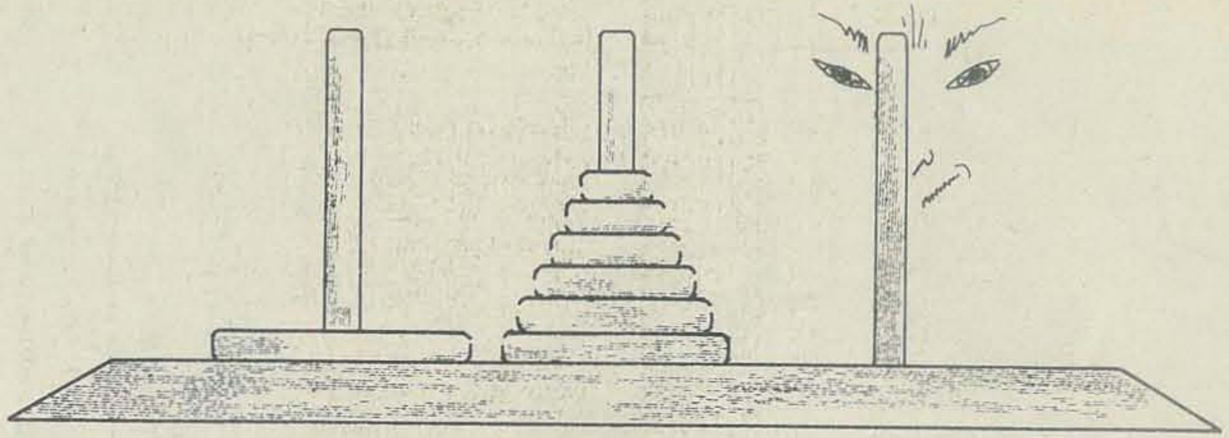
Swiftly the connections are made. Billy Basic is hooked up to one line of the emulator; F-Man to the other. Controls are set, pinouts adjusted, and voltage levels balanced. The final switch is thrown . . . .

Deep within the Stronghold, Linea waits impatiently with Angus for the results of the attempted transfer-and-reload function.




THE FURTHER ADVENTURES
OF
FORTRA
MA
Volume III
Episode 9
BY LEE SCHNEIDER & TODD V
Linea anxiously branches forward to
figure of her long-time friend. He stands beneath the great machine, still connected to the emulator outputs . . . .
from the source she expected.
What's going on???
Aye then . . . that be it. Me emulator is turned off. From here on they're running on their own!
Well, they *look* all right enough . . . .
Billy! Are you all right? Did the transfer complete successfully?
I . . . I feel fine, Linea . . . but what are you doing over *there* when I'm over *here*?
What's wrong, Linea? It's *me*, Billy Basic! Where's F-Man?
HUH??
I'm right here, Billy . . . and it worked! My memory is restored!
But what happened to *you*, Billy? You sure look *strange*.
Funny . . . you're looking sort of strange *too*, F-Man! What's going on?
Suddenly, like the rising edge of a fast clock, it dawns upon Linea just what has happened . . . .
Angus! they're *inverted*!
You've loaded them into each other's program space!!!
The elder McIntel checks his front-panel switch settings, and sure enough . . . .
Oops! Er . . . ye be excusin' me, folks. I seem to ha' gotten me Transfer Instruction Direction Control Bit reversed!
All right, all o' ye back under the Emulator, an' I'll be havin' ye back to normal quick as ye can say "Holy Macro!"
Oh no!
DRAWING BY AMIYA


A flip of the emulator switch and a brief ZAP later . . . .
Linea updates the recently-restored version of F-Man, filling in the vacant locations in his data array . . . .
But . . . then the fight has just begun!
Yes, and unless we get into execution soon, our efforts will be terminated. With our defeat, we will lose our beloved land to that villain the Glitchmaster and his noisy crew!
Aye . . . but how? The Glitchmaster's troops still hold the Core Plains between here and Capital City. To make a mass data transfer *around* the plains in so few cycles is impossible!
Then we *must* get our resistance forces to connect with the General's network immediately!
I have a plan, Angus . . . come along, all of you! And hurry. We have no cycles to lose!
Linea! Billy! That's *much* better. I feel like my old self again!
Tell me . . . what of the struggle to overthrow the Glitchmaster? Is the battle won?
On with the show . . .
Meanwhile, in a faroff-province, units of the Underground Resistance Movement engage themselves in the attack on Capital City, the very stronghold of the Glitchmaster and his high-level forces.
With unending propagation, they surge forward beneath the Underground banner. Through the hail of deadly voltage spikes, they repeatedly charge and discharge the capacitive guards that hold up the potential on the city walls.
Through the oscillations, the venerable old General Wirewound, who has a tolerance above them all, directs the battle. He makes adjustments to the lines as needed; shunting off his power in new directions at each turn.
Current losses are high, General. Half of our supplies have been shorted by the Glitchmaster's Deregulation Projectors. The rest are severely overloaded! We need more resistance . . . or we might as well give up and go ohm!
Down with the Upper Cases!
ZAP!
Freedom for the Land of the Little People!
Order the reserve charge carriers to the front junction lines . . . and prepare our RAM batteries for a sustained-data assault on the Main Gates!
ZAP!
Couldn't we terminate our resistance attack at this node and re-form our network to re-engage at a different branch?
No, we are totally committed. If we attempt to disconnect now the Glitchmaster's inductive voltage forces would follow our retreating current lines, and the transients would destroy us! We *must* continue the attack!
If only Linea were here . . .
A sudden Interrupt comes by way of a messenger from one of the remote segments.
General!
Look! There . . . over the Plains!
And what do they sense coming in through the Great Core Plains? Could it be the lines of supporting force that they are so direly in need of . . . or is it just another streak of bad flux?
I see it . . . but I don't believe it!
Wha . . . ?
It's . . . it's . . .
As all turn to behold the strange sight . . .
Will Linea and her resistance arrive in time? Will the evil Glitchmaster ever be banished forever from Microprocessorland? Will Billy Basic ever again RUN free in his native homeland?
For the answers to these and other uncompilable questions, tune in for the next episode . . . same timeout, same status flag.



TRS-80: TOWER OF HANOI

BY HERBERT L. DERSHEM

Several issues ago we published a couple of program examples using recursion. At that time, we asked if anyone had other home computer applications that were based on recursive procedures. Tower of Hanoi, as presented here, makes excellent use of a recursive technique for implementing the "move" section of the program.

But, beware!! If your micro does not have a recursive BASIC, this program will not work for you as it is written. You will have to alter lines 1000-1100. – RZ

This program is a version of the Tower of Hanoi puzzle, also commonly known as Donuts (see "Donuts for Kids" by Ron Sontore in *Calculators/Computers*, November, 1977). The puzzle consists of N doughnut shaped discs, of different diameters, and three vertical towers on which the discs can be placed. The problem begins with all the discs on the left-most of the three towers, called Tower 1. The larger discs are at the bottom of the stack.

The object is to move all of the discs from Tower 1 to Tower 2, the middle of the three towers. There are three rules to follow in moving the discs:

1) Only one disc can be moved at a time.
2) A disc is moved by removing it from the top of the stack of discs of one tower and placing it on the top of a stack of discs of another tower.
3) No disc may ever be placed on top of a smaller disc.

The program given here is written in Level II BASIC for the Radio Shack TRS-80. The user has the options of solving the puzzle himself or watching the computer solve it. In either case, the user has the choice of the number of discs to be used, up to a maximum of 15. The program also provides a time estimate for how long the solution will take. This is helpful in the case of 15 discs where at least 32,767 moves are required. When the computer solves the puzzle, the user has the option of controlling the length of time between moves. This feature is helpful for demonstration purposes.

The method used by the computer is recursive, which means that the routine to determine the moves calls upon itself. The basic principle here is that if you have a routine to successfully move n-1 discs from Tower i to Tower j, for any i and j, then you can move n discs from Tower i to Tower j by the following procedure:

1) Move n-1 discs from Tower i to Tower k.
2) Move 1 disc from Tower i to Tower j.
3) Move n-1 discs from Tower k to Tower j.

where k is the number of the Tower which is not Tower i or Tower j. In other words, k = 6 – i – j. This recursive procedure is shown in statements 1000-1100.

This program has been used with students at both the elementary and the secondary level. The puzzle is explained by showing the students the computer solution with three discs. Usually, about a three second delay is requested between moves so that an explanation of each move can be given. Following this, the students are asked to solve the puzzle with four discs and compare their number of moves to what the computer reports as optimum. After several tries or after they do obtain the optimum number of moves, they watch the computer solve the puzzle for four discs, and then go on to try it for five. After the optimum effort for five discs is found, the students are encouraged to conjecture on the effort needed for six or more. For a larger number of discs, running the program at least to the point where the time estimate is produced is also interesting.

TOWER LISTING

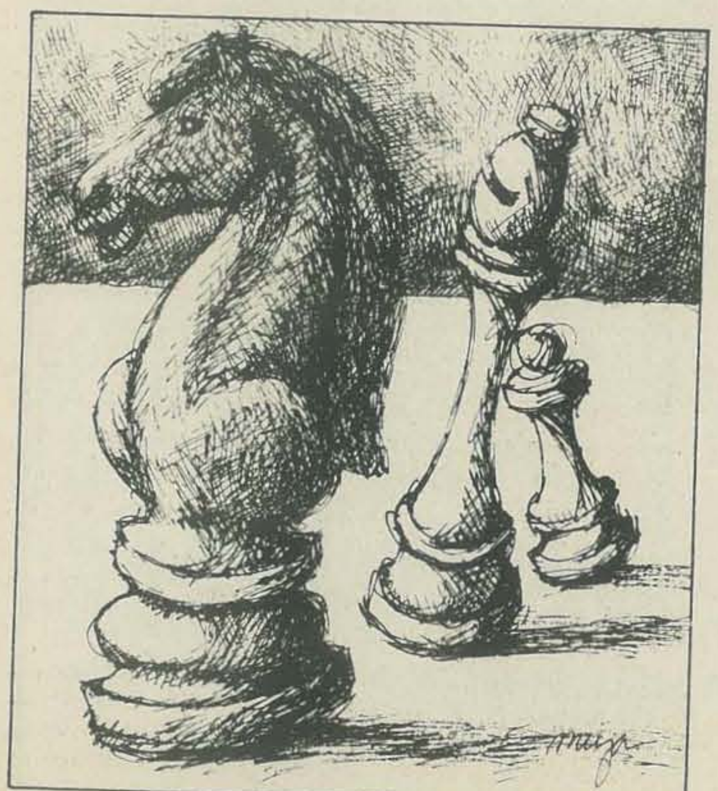
```
10 REM TOWER OF HANOI PUZZLE IN LEVEL II BASIC
20 KT=0: PRINT "TOWER OF HANOI"
30 INPUT "DO YOU WANT INSTRUCTIONS"; R$
40 GOSUB 4000
50 ON R GOTO 500,60,30
60 INPUT "HOW MANY DISCS DO YOU WANT"; ND
70 R=ND: LO=1: H1=15: GOSUB 3000
80 IF B=1 THEN 60
90 DIM N(3),D(3,ND),E1(ND),F1(ND),D1(ND)
100 INPUT "DO YOU WANT TO MAKE THE MOVES YOURSELF";R$
110 GOSUB 4000
120 ON R GOTO 300,130,100
130 INPUT "HOW MANY SECONDS DELAY DO YOU WANT BETWEEN MOVES";NS
140 R=NS: LO=0: H1=10: GOSUB 3020
150 IF B=1 THEN 130
160 T=(2↑ND-1)*(NS+.57)
170 GOSUB 6000
180 D=ND: E=1: F=2
190 GOSUB 5000
200 GOSUB 990
210 FOR I=1 TO 100: PRINT 896, "(APPLAUSE! APPLAUSE!)";
220 FOR J=1 TO 100: NEXT J: PRINT 896,CHR$(30);
230 FOR J=1 TO 100: NEXT J: NEXT I
240 STOP
290 REM USER MAKES MOVES.
300 T=(2↑ND-1)*5.57
310 PRINT "ASSUMING IT TAKES YOU 5 SECONDS PER MOVE"
320 GOSUB 6000
330 D=ND: E=1: F=2
340 GOSUB 5000
350 PRINT@768,CHR$(30);: PRINT@768,"ENTER MOVE";KT+1;": FROM"
360 INPUT E
370 R=E: LO=1: H1=3: GOSUB 3000
380 IF B=1 THEN 350
390 PRINT@768,CHR$(30);:PRINT @832, CHR&(30);
400 PRINT@768,"ENTER YOUR MOVE";KT+1;": FROM";E;"TO";
410 INPUT F
420 R=F: GOSUB 3000
430 IF B=1 THEN 400 ELSE PRINT@832,CHR$(30);
440 (E=F) OR (N(E)<=0) OR (N(F)<>0 AND D(E,N(E))>D(F,N(F)) THEN PRINT@832,
      "ILLEGAL MOVE";CHR$(30);: GOTO 350
450 GOSUB 2000
460 IF N(2)<ND THEN 350
470 IF KT>2↑ND-1 THEN PRINT@896,"YOU DID IT, BUT YOU COULD HAVE DONE IT IN";
      2↑ND-1; "MOVES.";
480 IF KT=2↑ND-1 THEN PRINT@896,"CONGRATULATIONS! YOU DID IT IN THE LEAST
      POSSIBLE MOVES.";
490 FOR I=1 TO 10000: NEXT I: STOP
500 REM PRINT INSTRUCTIONS
510 PRINT "YOU ARE GIVEN THREE TOWERS ON WHICH YOU CAN PLACE DISCS WITH"
520 PRINT "HOLES IN THE CENTER. THE DISCS ARE ALL ON TOWER 1 TO START"
530 PRINT "WITH THE LARGEST ON THE BOTTOM AND THE SMALLEST ON TOP.THE GOAL"
540 PRINT "IS TO MOVE THE DISCS ONE AT A TIME SO THAT ALL DISCS END UP ON"
550 PRINT "TOWER 2. THE ONLY RESTRICTION IS THAT A LARGER DISC MAY NEVER"
560 PRINT "BE PLACED ON A SMALLER ONE. YOU HAVE THE OPTION OF"
570 PRINT "SOLVING THE PUZZLE YOURSELF OR ASKING THE COMPUTER TO SOLVE IT."
580 PRINT "GOOD LUCK!"
590 GOTO 60
970 REM SUBROUTINE TO RECURSIVELY MOVE D DISCS FROM TOWER E TO TOWER F.
980 REM FIRST ENTRY IS TO LINE 990. SUBSEQUENT CALLS ARE TO LINE 1000.
990 S=1
1000 IF D=1 THEN GOSUB 2000
1010 IF D=1 THEN 1090 ELSE G=6-E-F
1020 E1(S)=E: F1(S)=F: D1(S)=D
1030 S=S+1: D=D-1: F=G
1040 GOSUB 1000
1050 E=E1(S): F=F1(S): D=D1(S)
1060 GOSUB 2000
1070 S=S+1: D=D-1: E=6-E-F
1080 GOSUB 1000
1090 S=S-1
1100 RETURN
2000 REM MOVE A DISC FROM TOWER E TO TOWER F.
2010 KT=KT+1: B=ND+10: PRINT@0,STR$(KT);".MOVE FROM";E;"TO";F;
2020 N(E)=N(E)-1:N(F)=N(F)+1
2030 SZ=D(E,N(E)+1)
2040 D(F,N(F))=SZ
2050 XE=40*E-20
2060 XF=40*F-20
2070 FOR J=-SZ-1 TO SZ+1
2080 IF J<>0 THEN RESET (XE+J,B-N(E)): SET(XF+J,B-N(F)+1)
2090 NEXT J
2100 FOR J=1 TO 500*NS: NEXT J
2110 RETURN
3000 REM CHECK RESPONSE R TO SEE IF IT IS BETWEEN LO AND H1.
3010 IF R<>INT(R) THEN PRINT "RESPONSE MUST BE AN INTEGER": B=1:RETURN
3020 IF R<LO THEN PRINT "RESPONSE MUST BE NO LESS THAN";LO: B=1:RETURN
3030 IF R>H1 THEN PRINT "RESPONSE MUST BE NO MORE THAN";H1: B=1:RETURN
3040 B=0: RETURN
4000 REM CHECK RESPONSE R$ FOR YES OR NO.
4010 IF R$="YES" THEN R=1: RETURN
4020 IF R$="NO" THEN R=2: RETURN
4030 PRINT "PLEASE ENTER YES OR NO.": R=3: RETURN
5000 REM DRAW TOWERS
5010 CLS
5020 L=INT((ND+10)/2)*64-64
5030 FOR I=1 TO 3
5040 PRINT@L+20*I-12,"TOWER";I;
5050 NEXT I
5060 PRINT@L+150,"TOWER OF HANOI";
5070 FOR I=1 TO ND
5080 D(1,I)=ND+1-I
5090 NEXT I
5100 N(1)=ND: N(2)=0:N(3)=0
5110 B=ND+10
5120 FOR I=6 TO B
5130 SET(20,I): SET(60,I): SET(100,I)
5140 NEXT I
5150 FOR I=B TO B-ND+1 STEP-1
5160 W=ND-B+I-1
5170 FOR J=20-W TO 20+W
5180 SET(J,I)
5190 NEXT J
5200 NEXT I
5210 RETURN
6000 REM PRINT TIME ESTIMATE
6010 PRINT "TIME ESTIMATE FOR THIS PUZZLE:";
6020 D=0: H=0: M=0
6030 IF T>=86400 THEN D=INT(T/86400): PRINT D; "DAYS";
6040 T=T-D*86400
6050 IF T>=3600 THEN H=INT(T/3600): PRINT H; "HOURS";
6060 T=T-H*3600
6070 IF T>=60 THEN M=INT(T/60): PRINT M;"MINUTES";
6080 T=T-M*60
6090 PRINT T; "SECONDS".
6100 FOR I=1 TO 1500: NEXT I
6110 RETURN
```

a New Algorithm for Chess

David Watters indicated in the letter that came with this part of their chess article that David Chelberg is away in Russia. So, for the next few months, you can contact the D&D chess team at: 903 We-Go Trail, Mt. Prospect, IL 60056. Perhaps David C. will return from Russia with several new topics for this continuing series of articles. I don't mind as long as the manuscripts are not covered with beet juice and vodka stains. How about a parapsychological pawn to the Q3? Now that would be interesting! –RZ

PART IV: MIDDLE GAME STRATEGY – THE DYNAMIC EVALUATION

BY DAVID CHELBERG & DAVID WATTERS



In our last article, we began the discussion of our middle game strategy. We discussed the move generation routine that prepared the data for the computer to make its analysis. In this analysis, each possible move is tested for a variety of features. For each feature, the move is given a value. A cumulative sum is obtained and the move with the highest value is chosen. The aspects of each move that we feel are important can be classified into two categories: dynamic evaluation and static evaluation. The static evaluation concerns itself with long range problems such as piece development, centralized attacks and king defense. The dynamic evaluation is for the short-term problems such as pieces in danger, pins and counterattacks. This article deals with the dynamic evaluation of the middle game. Again, we stress that it is helpful to have read our previous articles before reading this one.

VITAL PROTECTORS

The major concern of the dynamic analysis is—pieces in danger. Since our program is designed to follow human thought processes, let's examine what a human player might consider in making his move. Probably the first consideration that comes to mind in response to an opponent's move is "How does that move threaten me?"; followed by "How can I avert this threat?" Later, the player might consider how he can best paralyze his opponent, but the first test he must make is to determine which moves *cannot* be made because of danger to other pieces. This section of our program is called "Vital Protection."

In this routine, a search is made for computer pieces under attack. When a piece is found, the A-matrix is employed to find the attacker. For a piece under attack, two cases exist: underprotection and overprotection. Underprotection occurs when the advantage is for the opponent to make the capture. In this case, the "protectors" are free to move as they please since the opponent will make the capture regardless of their presence. Underprotection is examined more closely in a later section.

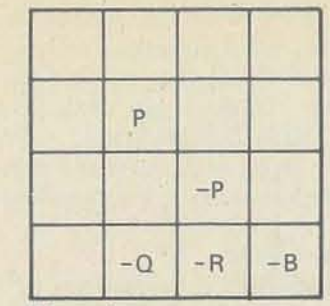


Figure 1
None Vital

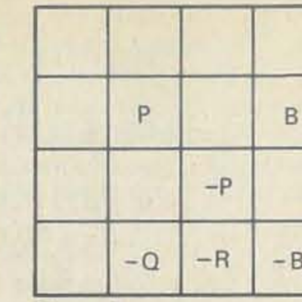


Figure 2
Bishop Vital

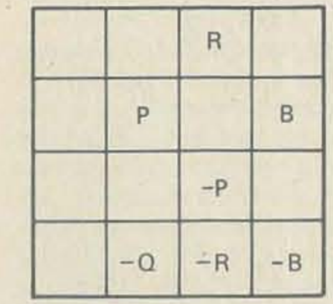


Figure 3
All Vital

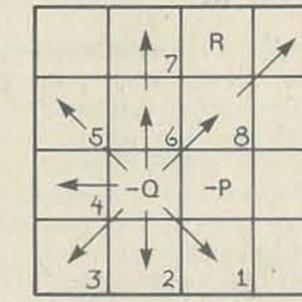


Figure 4
Mobility of a Vital Protector

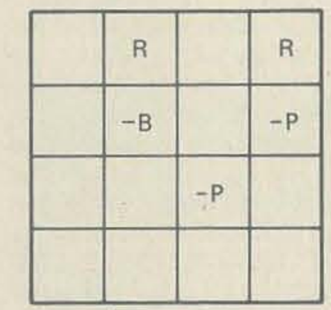


Figure 5
Double Vital Protection. The pawn only protects the bishop.

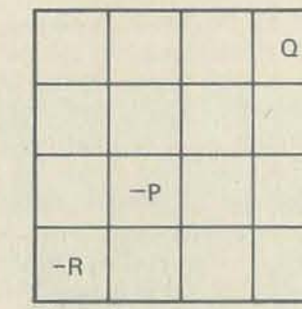


Figure 6
Pin/hurdle predicament

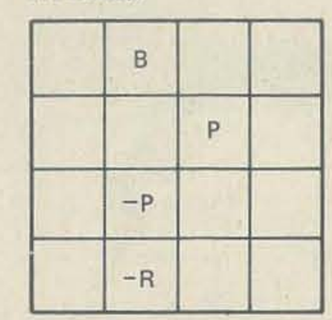


Figure 7
Indirect Attack via P X P

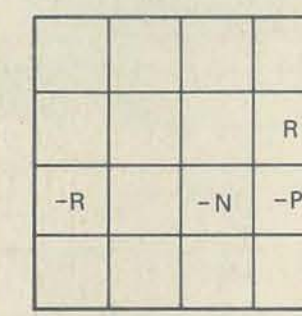


Figure 8
Indirect Protection Move Knight

The case where it is not advantageous for the opponent to make the capture is known as overprotection. In this case, some, none or all of the protectors may be vital to the welfare of the attacked piece. (See Figures 1, 2 and 3.) Thus, it becomes necessary to determine which pieces are vital protectors, and which are not. To do this, we implement the "capture sequence" (described below) to determine whether the opponent would make the capture in the absence of one protector. Those protectors, which in their absence create a capture advantage for the opponent, are the vital protectors. Those pieces are the subjects of this section of the program.

The possible moves of the vital protectors are closely scrutinized. If a move jeopardizes the attacked piece, it is given a negative value equal to the net loss of material in the ensuing trade. In Figure 4, the queen has limited mobility – if she moves to squares 3,5,7 or 9, she can no longer protect the pawn. Moves to those squares would be given a negative value on that basis. Moves such as these are also "black-listed" for future reference. The black-list is a column matrix containing the coordinates of the attacked piece, the vital protector, the square to be moved to, as well as the net loss that would be incurred as a result of this move. If, as in Figure 5, a piece is vital to the protection of two pieces, it is considered vital only to the larger valued one (bishop), leaving the lesser piece (pawn) underprotected. Moves which were black-listed because of the lesser piece are restored to prior standing. Upon finding a vital protector, all of the remaining computer's moves are examined. If a move blocks the path of this vital protector, and is incapable of protecting the attacked piece, it is given a negative value and black-listed.

In general, vital protectors are pieces which should not be moved. If a vital protector is used to attack a more valuable piece, the computer's own defensive position is often compromised. It is better to relieve the pressure on the piece by adding another protector before the attack is made.

Another case in which a piece should not be moved occurs when a piece is either pinned or hurdled. In this section of the dynamic evaluation, the whole board is examined. Figure 6 shows the pin/hurdle predicament. When a piece such as the pawn is moved forward, the rook becomes subject to an attack by the queen. This is a poor move, and a negative value is assigned.

On the other hand (Figure 7), the pawn move could provide an opportunity for attack by the computer. This move is potentially good. However, it is left to the "attack" routine to determine how good. No value is assigned because the capture is not imminent, and the situation will probably change to prevent the attack. Basically, though, these two sections catalog those moves which are undesirable because the pieces are needed elsewhere. By having this section as the first major consideration of our dynamic evaluation, many disasters have been averted.

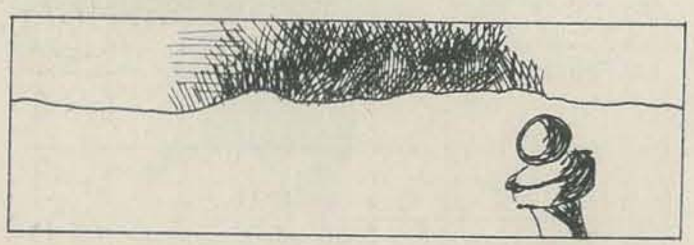
CAPTURE SEQUENCES

Now that the catalog of explosive moves has been established, each move can be examined on its own merit. For this we employ our "capture sequence." Here, all possible moves are made hypothetically. From the square that is occupied by the new move, the A-matrix is used to make two column matrices. One contains all the computer's pieces protecting the square, and the other contains the attackers. The opponent moves first, capturing the piece whose move is under consideration. The computer recaptures until the position is dead—dead means that for one side or the other, to continue the exchange would be unprofitable. The initial move is then given a value equal to the net loss or gain of the exchange. This section evaluates the relative safety of each square on the board. It shows the direct implication of a particular move. The previous section shows the indirect implications.

The next part of the dynamic evaluation is preparatory. All attacks both direct and indirect must be found. They will be used in the next two sections – protecting the computer's endangered pieces and thwarting the opponent's most likely future attacks.

In finding the attacks, the basic question asked is "by moving to this square, can any opponent's piece be attacked?" The variation to this (which has already been explored) is "by moving this piece, can another piece be freed to attack the opponent?" These latter moves have already been generated and it remains to be determined whether the attack is formidable or not. To find the attacks, a mini-move generation routine is implemented. A list is compiled of all the potential attacks (direct and indirect). This list is pruned by determining the usefulness of each attack. An attack is considered useful if either the attacked piece is unprotected or the attacker is of a lesser value than the attacked piece.

After the list has been pruned, it is scrutinized for king attacks. If the computer can put the opponent in check, the move is made hypothetically, and a test for checkmate is implemented. If checkmate is possible, the move is made immediately. If the opponent can checkmate the computer on the next move, the prevention of this move is given priority. The reason the attacks are compiled here, rather than in the move generation routine is because of an earlier cut off. In the preceding routine, if an excellent capture is found, it is assumed to be the best move (usually this is a response move in a trade) and it is immediately made. Attacks are not needed for that consideration, and a great deal of time is saved by postponing their generation.



PROTECTION

Probably the most important and extensive portion of the dynamic evaluation is the section which we call "Protection." This routine deals with underprotected pieces and how to rectify their condition. It begins by finding a piece which is in imminent danger. It proceeds through a capture sequence to determine the net loss involved as well as the largest valued piece that can successfully protect the endangered piece. Moves that remove the threat of attack are given a raw positive value equal to the net loss saved by making that move. There are of course many ways to rescue the piece in danger. The most common method is through direct protection. All possible moves are candidates, although those black-listed as vital protectors are avoided. They are only permitted to save a piece of greater magnitude than the one which they already protect. The value assigned is the net loss saved, less the net loss of the old piece that would now be jeopardized. Throughout the protection routine, vital protectors are treated in this manner.

A variation to direct protection is indirect or discovered protection. It is realized by moving a piece out of the path of a potential protector. (See Figure 8.) Other ways to circumvent attack are: moving the endangered piece, capturing the attacker, pinning the attacker to the king, interposing a lesser piece, and counterattacking. Again these moves are given a raw value equal to the net loss saved. In no case is a move given a double value. That is pinning the attacker and protecting the endangered piece at the same time is given only one value.

RELIABILITY OF STRATEGIES

All of these methods can successfully save the endangered piece. However, we feel that some methods are more reliable than others. Therefore, the raw value given to a move is adjusted slightly so as to give priority to those moves which are more dependable. Direct protection is probably the best method, because it usually strengthens the overall defensive structure. Indirect protection is a close second, but not as consistent as direct protection because the protector is usually too far removed, too powerful a piece, and easily open to attack. The next best method is to capture the attacker. This method is not as good as the former ones in general because it tends to weaken the general defensive structure as well as decentralize the offense.

Moving the piece in danger is the next most viable alternative. This alternative usually causes a loss in the offensive edge, because most safe moves lie in a retreating direction. In addition, this alternative has in some cases led to a draw by repeated position. Thus it is discouraged to some degree. The next best method is to interpose a lesser piece. This is the last method that decisively ends the attack. It is not necessarily desirable, since it creates a pin situation. Pinning the attacker to the king only postpones the threat since the opponent can usually remove the pin on the next move by castling or interposing a piece. This move often strengthens the opponent's defensive position.

The method of last resort is the counterattack. This delay tactic is only helpful if the piece the computer uses in the counterattack prepares a way of escape for the endangered piece on the next move. The major disadvantage of a counterattack strategy is that one can end up with two pieces in danger instead of one. Of course this dilemma can only be resolved by another counterattack—leading to three threatened pieces, etc. So, a ranking procedure for protective efforts was implemented. The protection routine answers the question "how can I avert the threat?" If all the moves which are intrinsically bad have been weeded out, this last routine can be successfully implemented.

LOOKING AHEAD

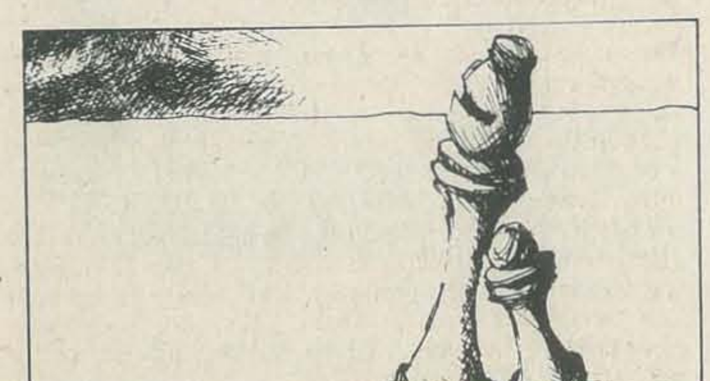
The last portion of the dynamic evaluation is the "plan ahead" stage. It is this section that is used to avert one-move mates. The opponent's attacks that were previously compiled are evaluated. Evasive action is taken. Many of the methods used in the protection routine, such as interposition and pinning the attacker, are used here in an attempt to avoid the ensuing attack. Since these attacks are only hypothetical, the moves that defy them are given small values – but an ounce of prevention is worth a pound of cure. If a potential problem can be solved when the time is available, the offensive edge can be maintained.

SUMMARY

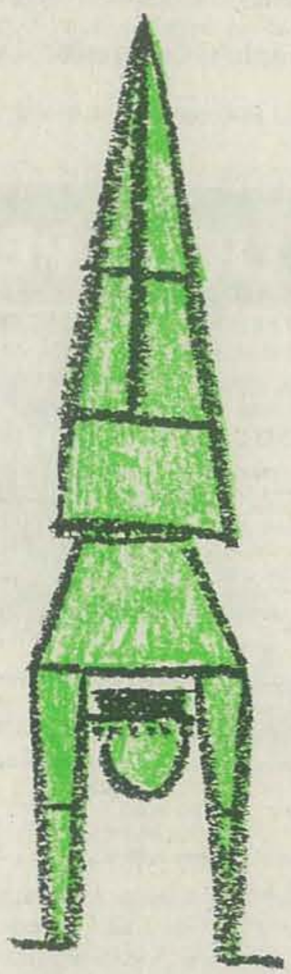
These are the basics of our dynamic strategy. To summarize: The first step is to find the vital protectors and pinned pieces. Next, evaluate each possible move from a material standpoint, using a capture sequence. A cutoff exists for unique outstanding captures. Next, all attacking moves are compiled and a search for one-move mates is implemented. Then, endangered pieces are protected, and finally, future attacks are prevented.

After finishing the dynamic evaluation, the computer begins its static evaluation. In general, the dynamic evaluation gives the static evaluation approximately five good moves (although all moves are evaluated in case any surprises exist). The static evaluation then chooses the best of these five or six moves on the basis of long-range goals. When no advantageous move is found in the dynamic evaluation, the static evaluation makes the total decision. This case might arise in the late opening.

In evaluating the efficiency of the dynamic evaluation, it was found that the portion which took the longest time was the last section—avoiding attacks. This section has the largest amount of data to process. On a time/move basis, protection takes the longest because of its multi-faceted evaluation process. In the overall picture, the dynamic evaluation takes a little more than one quarter of the time per move, with the move generation taking close to half and the static evaluation taking the remainder of the time. Generating the attacks is considered to be a part of move generation as its function is indeed generative rather than evaluative. Our next article will deal with the static evaluation and perhaps a skeletal description of end-game strategy. Again, if you have any questions concerning our program, feel free to write us.



Texas Instruments graphics & animation Part 1



BY DON INMAN

Here is a second installment in our ongoing introductory series on the new TI 99/4 Home Computer. In this article Don covers the basics of creating a graphics symbol within TI BASIC.

He promises to make Mr. Bojangles dance in the next piece. Now if we could only have Mr. Bojangles sing while he dances – wouldn't that be clever . . . ? – RZ

One of the advantages of the home computer over the number-crunching monstrosities of the past is the ability to produce graphics on the video screen. With the addition of color and sound, you may someday be producing your own movies created on your friendly home computer. Why not animated cartoons in full color and sound?

In the Sept-Oct issue of *RC*, we presented some of the sound capabilities of the TI 99/4 Home Computer. In this and future issues, we'll look at some of the graphic capabilities.

The material presented here is based on *Introduction to TI BASIC*, a book to be published soon by Hayden Book Company, Inc.*

Many different methods are being used by various manufacturers to display non-standard symbols on the video screen. These non-standard symbols are then combined to produce a "picture" (a graphic representation) of some real or imagined object.

Standard typing characters can be used in combinations to produce pictures, but the results are not very satisfactory. Look at the following example.

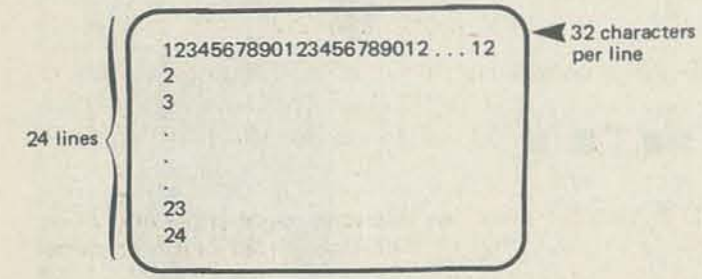
```
        X X X
    X           X
          _ _
 (        0 0        )
           L
    X           X
          I I I
      X       X
         X X X
```

Attempts to "animate" such pictures with the computer are almost impossible. However, TI has provided a way to modify the method of printing standard typing symbols so that you can create symbols of your own. First, let's take a look at how a standard symbol is created.

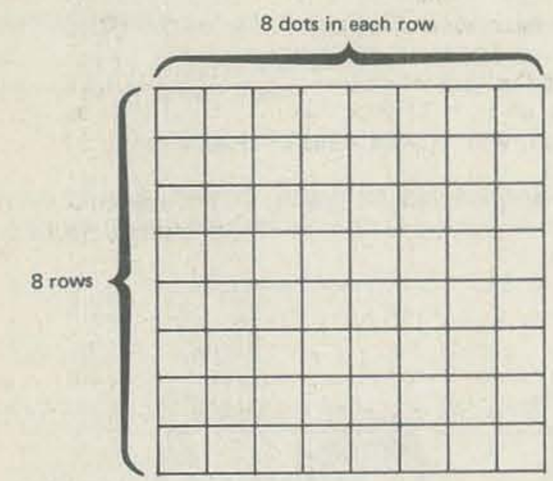
* *Introduction to TI BASIC*; Inman, Zamora, Albrecht, and Dymax; Hayden Book Company, Inc., 50 Essex St., Rochelle Park, N.J. 07662.

STANDARD CHARACTERS

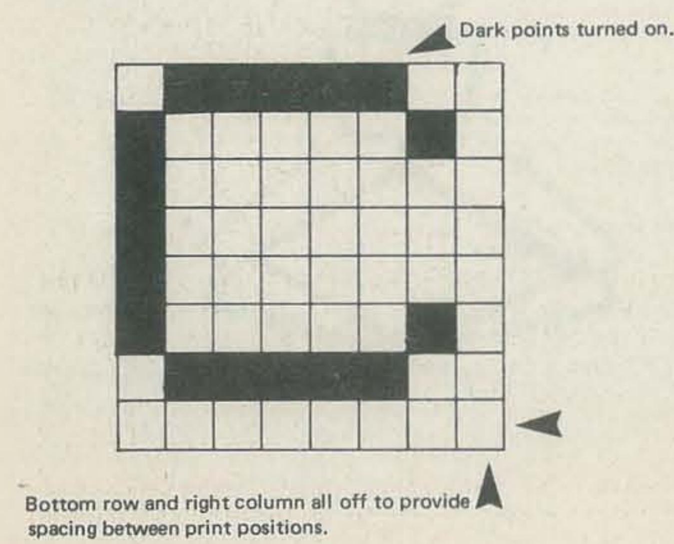
The computer prints standard characters on its TV monitor in 24 lines composed of 32 characters per line.



Now try to visualize *one* of the individual printing positions. It is made up of 64 tiny dots or areas arranged in 8 rows with 8 dots in each row.



A standard character is made by turning on specific dots to form the desired character. Suppose, for example, that you type the letter C. The following pattern of dots might be turned on to form the letter C.

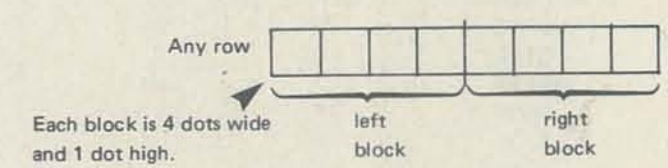


Each standard character that can be displayed on the video screen has its own unique pattern of "turned on" dots. This pattern is predetermined and fixed on the TI computer. However, there is a way for you to create your own patterns.

NON-STANDARD CHARACTERS

A character on the screen, either a standard character or one that you create, is formed by the dots within the 8 by 8 grid discussed above. A character is created by turning some dots on and leaving others off. To create a new character, you must tell the computer which dots to turn on and which dots to leave off.

Think of each row of the grid as being divided into two blocks.



A Dot Code is used for each block to tell the computer which dots to turn on. If a dot is to be turned on, a 1 is used in the corresponding cell within a block. If a dot is to be left off, a 0 is used in that cell.

Example:

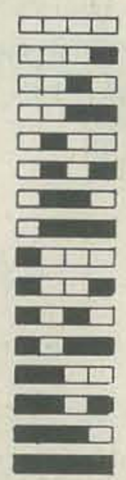
For one block of one row of the 8 by 8 grid:

0	1	0	1

left block — Turn these two "cells" in the left block of this row ON. Leave all others off.

Here is a table of all the possible Dot Codes and the patterns that can be produced.

Dot Code	Block Pattern
0000	
0001	
0010	
0011	
0100	
0101	
0110	
0111	
1000	
1001	
1010	
1011	
1100	
1101	
1110	
1111	



Those of you who are familiar with the conversion of binary numbers to hexadecimal numbers will recognize that the Dot Code (of binary numbers) can be written in a Shorthand Code of hexadecimal numbers. The Dot Codes are converted to a Shorthand Code which can be used by the computer. This conversion is shown in the following table.

Dot Code	Shorthand Code
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9
1010	A
1011	B
1100	C
1101	D
1110	E
1111	F

Notice that letters are used for these. (A–F)

To fill up one row of the 8 by 8 character grid, you must use two Shorthand Codes, one for the left block of 4 cells and one for the right block.

Examples:

To produce this row, use Dot Codes

0110 0101

6 5 ◄ Shorthand Code

left block, right block

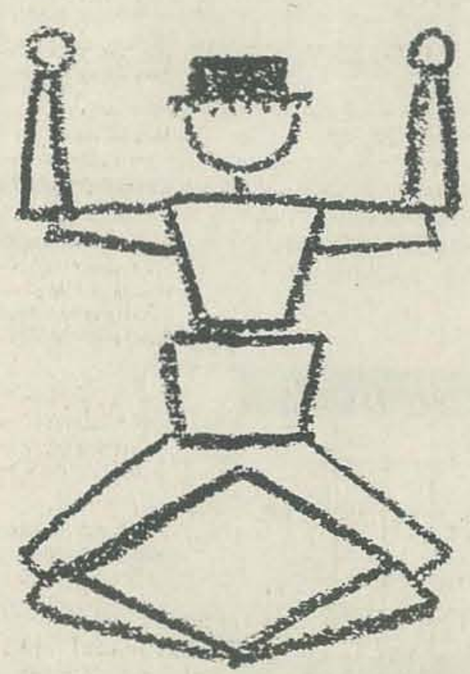
To produce this row, use Dot Codes

1010 0001

A 1 ◄ Shorthand Code

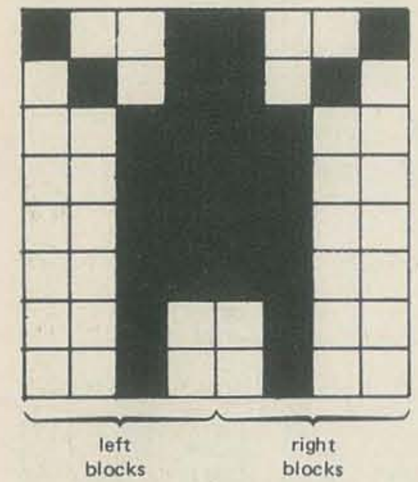
left block, right block

To produce a complete 8 by 8 grid, we need to design 8 of these pairs of Shorthand Codes, one for each row of the print position.



Example:

To produce this pattern: use Shorthand Codes



row 1	99
row 2	5A
row 3	3C
row 4	3C
row 5	3C
row 6	3C
row 7	24
row 8	24

To combine all the 8 rows together to make the complete picture, you combine the Shorthand Codes into one long string, row by row.

LET A$ = "995A3C3C3C3C2424"

Now let's write a short program using TI BASIC to put our "picture" on the screen.

10 REM** LITTLE PERSON**	
20 CALL CLEAR	This clears the screen.
30 LET A$="995A3C3C3C3C2424"	The Shorthand Codes define the picture.
40 CALL CHAR (96, A$)	This replaces the character (96) with our created character (A$).
50 CALL COLOR (9, 2, 16)	Sets the colors: black (2) and white (16). It also gives the character set of the redefined character (9).
60 CALL VCHAR (12, 16, 96)	Defines the row (12) and column (16) where the picture (96) will appear on the screen.
70 GOTO 70	Holds the picture on screen for viewing.

This program will put the little guy on the screen. But how do you make him move? Tune in to the next issue of *Recreational Computing,* and we will teach Mr. Bojangles, our little guy, to dance for you.

Workshops Beat Math Anxiety

BY STANLEY KOGELMAN

Dr. Kogelman is director of Mind Over Math, 40-18 209th St., Bayside, Queens, New York 11361. Whew! I'm glad that's out of the way. I was beginning to get anxious over the size of that address.

Our modern world is filled with many stress-producing and anxiety-generating objects, ideas, and processes. We are buffeted by the media, the pressures of work, and often our own imaginations. So what are we to do? Well, there are de-stressing workshops, de-conditioning processes, hypnosis, and the stuff of this article—Math Anxiety Workshops.

This article, the first in what we hope to be a series on the issues of anxiety and technology, is a good starting point for a lot of us. Dr. Kogelman specializes in helping people overcome their anxieties related to mathematics. In the future, we plan to extend the scope of these articles to include the anxieties that computers are bringing into people's lives. So, relax, get comfortable, and go ahead and read . . What? You're getting nervous? Oh, come now! There is a number at the bottom of the page . . . but you're not being asked to add anything to it. It is really o.k. . . . just relax and read . . . —RZ

Being a mathematician does not win friends at a cocktail party. Typically, the reaction to learning I am a mathematician is "Math?!! I always hated it. It was my worst subject. Excuse me, I'm going to get a drink." Now, however, when I say I specialize in overcoming math anxiety, the response is far friendlier. They say, "Math anxiety?!! I've got it. All you have to do to see me panic is say, 'If two men can dig a ditch . . .' I remember in seventh grade . . ."

Reprinted by permission. © 1979 by *SIAM NEWS*, Vol. 12, No. 2, April 1979.

Most people who have difficulty with math believe they "just can't do it." Their immediate reaction to math is so intense that their memory, attention and concentration become severely impaired. Otherwise intelligent and successful people find they are just unable to learn or do math. You often see them nodding "yes" when in fact you know they want to say "no!" Even the most supportive and patient tutoring proves ineffective.

After years of teaching and doing mathematical research, I returned to school to earn a master's degree in clinical social work. While at Smith College, I did a research project aimed at discovering causes of math anxiety, which prevents people from coping with anything that has numbers. For the last three years, I have been offering Mind Over Math workshops which enable people to overcome debilitating anxiety so that they are free to learn mathematics.

Two-thirds of the people who attend the workshops are women. While research indicates that women avoid math more than men, my belief is that there are probably equal numbers of men and women who could be called "math anxious." The difference is that women too easily say, "I can't do it," while men, with much bravado, say, "I could do it if I tried." Nevertheless, both may experience the same sense of blocking and inadequacy.

Math anxiety does not have a single cause, but results from any of several factors reinforced by bad experiences. The three most dominant factors are:

Women, especially in junior high school, are affected by role expectations. Math is something boys are supposed to do. Many girls are led to believe that it is unfeminine to excel in mathematics and therefore reject it. And they may meet little opposition to dropping math at the earliest opportunity.

Both men and women often perceive math as being a rigid, authoritarian subject consisting primarily of a set of rules to be obeyed. Math books contribute to the rigid image by the formal style in which they are written. To many, this appears inflexible and forbidding. It is as if math were something that has always been there, not something men and women discovered and developed.

While the human expression in mathematics comes through the personal ways math is conceptualized and problems are solved, these qualities are rarely conveyed in primary and secondary schools. The emphasis there is on communication of a specific body of knowledge through lectures and drills. Individuality and intuition are not encouraged.

The focus on detail and symbolism required by math can be very unpleasant, frustrating, and annoying to those who hate details of any kind. The alternative to concentrating on detail is to get an overall sense of a subject and rely primarily on feelings. In math books, however, there is no way to get a sense of the concepts without careful reading.

There are two contrasting styles of thinking: convergent and divergent. The convergent focuses stepwise to a point, while the divergent is more "creative," free-flowing, imaginative. While math requires both styles of thinking, few realize this, and a primarily divergent thinker may resent having to obtain a definite answer by following a formal sequence of steps.

THE MYTHS

Many of the feelings people have about the cold, logical, rigid qualities of mathematics are related to firmly entrenched and generally believed myths about the subject. The twelve math myths are: Men are better than women; math requires logic, not intuition; you must always know how you got the answer; math is not creative; there is a best way to do a math problem; it's always important to get the answer exactly right; it's bad to count on your fingers; mathematicians do problems quickly in their heads; math requires a good memory; math is done by working intensely until the problem is solved; some people have a "math mind" and some don't; there is a magic key to doing math.

Demystification of mathematics is an essential part of overcoming math anxiety. Few non-mathematicians are aware that intuition is the cornerstone of doing math. Traditionally, mathematicians throw away their scrap paper and exhibit only the refined solution. The student, observing his instructor effortlessly solving a problem, thinks, "I could never do that." It doesn't occur to him that he is observing the result of hours of effort or years of practice.

Students are always surprised to learn that there were times when I had difficulty with mathematics, that I didn't solve all problems instantly in my head and that taking breaks helps.

The reactions of the most anxious students, while different in degree, became more comprehensible to me when I recalled the feelings I had when learning

mathematics. For example, I recalled the initial difficulty mastering ϵ . δ proofs, the confusion of the first course in math analysis and topology. When faced with new and difficult concepts, the initial reaction is a kind of fogginess in the head that is hard to see through, wondering if you are going to be able to understand, feeling like your mind is just not working, wondering if it is beyond you. The difference is that the math-anxious give up after a single try, thinking, "I knew I didn't have a mathematical mind," while the mathematician knows that this is just part of the process of learning math.

THE DEMYSTIFICATION PROCESS

Mind Over Math workshops aim at enabling the most anxious to overcome their fear of math. During five, two-hour sessions, 10-15 participants first talk about their experiences with math and then examine their approaches to solving simple problems. Gradually, they work through their negative attitudes until, finally, they can approach math as they would anything else. To avoid a classroom atmosphere, workshops meet in a lounge.

The first meeting is devoted to discussion of experiences and of math myths. At the next meeting, a description of daily activities like shopping, buying gas, and eating in a restaurant is handed out. This description includes a great many numbers in order to stimulate reactions to math which the group then discusses. While no specific problem is stated, participants are asked to make up a problem they can solve from any part of the story. Most striking is the attraction people have to what they can't do. If a problem is easy, they feel it is not worth doing. However, when they saw the sentence, "I had to get 4.8 gallons of gas," they began to freeze and go blank because they "knew" they couldn't do decimals. Understanding this reaction enables participants to begin to control it.

Through the next two sessions, participants receive more materials and find it getting easier and easier to read and work with combinations of words and numbers. They also begin to more realistically assess what they do and don't know. Although math as such is not discussed, many suggestions on how to study math are offered. There is particular emphasis on how to approach word problems as a translation and on how to develop insight into difficult problems by constructing simpler models.

The last session is devoted to discussing changes in outlook, underscoring growth and making recommendations for further study of mathematics based on individual needs.

Most people do not aspire to be mathematicians. They generally set realistic goals for themselves and just want to be comfortable with the math of everyday life and want to feel that they can learn whatever mathematics they need for their work. Usually this does not go beyond needing to develop skill with arithmetic, algebra, statistics and, perhaps, finite math and calculus.

I have yet to encounter anyone who could not attain his or her goals in math once the anxiety was overcome.

card no 898 **TRS-80** no 3347

B ◊ I ◊ N ◊ G ◊ O

BY KAREN D. CHEPKO

This program is Karen's second submission to RC and possibly the second program she has ever written. Not bad, Karen! Not bad at all. RND, POKE, SET, POINT, INKEY$, PRINT USING, and much more. . . . A micro-lesson in the use of many of the more advanced features of the TRS-80. I know we will be hearing more from Ms. Chepko. – RZ

There should be a requirement that each hobby computer have a label with the message: WARNING--This Product May Become Habit Forming! I am having a great time working on programs. My latest effort displays all of the possible numbers that can be called during a BINGO game. The program "calls the game" by selecting a random number with its letter. My only problem with this program was how to keep from choosing the same number twice. The solution to the problem also gave me a way to check BINGO's. As each number was selected, a location to the left of it is set on the screen. As the computer picks another number, it points at that screen location. If the location is SET, another number is then selected. At the end of the game, a quick review of the display tells me which numbers were called!

```
100 "BINGO BY KAREN CHEPKO"
110 ' THIEF RIVER FALLS, MINNESOTA, 56701
120 CLS : RANDOM
130 DEFINT A-Z
140 CLS
150 'SETS UP NUMBER DISPLAY
160 A=1 : B=15 : GOSUB 400
170 PRINT : A=16 : B=30 : GOSUB 400
180 PRINT : A=31 : B=45 : GOSUB 400
190 PRINT : A=46 : B=60 : GOSUB 400
200 PRINT : A=61 : B=75 : GOSUB 400
210 POKE 15423,66
220 POKE 15551,73
230 POKE 15679,78
240 POKE 15807,71
250 POKE 15935,79
260 'SELECTS AND DISPLAYS RANDOM NUMBERS
270 N=RND(75) : GOSUB470
280 IF POINT(X,Y)=-1 , GOTO 270
290 SET (X,Y)
300 PRINT@708,"TYPE 'N' FOR NEW NUMBER, ";
310 PRINT"'G' FOR NEW GAME, OR 'E' TO END."
320 PRINT@862,Q$;N;""
330 K$=INKEY$
340 IF K$="" , GOTO330
350 IF K$="N" , GOTO270
360 IF K$="G" , RUN 120
370 IF K$="E" , CLS : END
380 GOTO330
390 'SUBROUTINE FOR PRINTING NUMBERS
400 FOR C=ATOB
410 U$="####"
420 PRINT USING U$;C;
430 NEXTC
440 PRINT
450 RETURN
460 'SUBROUTINE FOR DISPLAYING RANDOM NUMBERS
470 IF N<16 , Q$="B" : GOTO520
480 IF N>15 AND N<31 , Q$="I" : GOTO 530
490 IF N>30 AND N<46 , Q$="N" : GOTO 540
500 IF N>45 AND N<61 , Q$="G" :GOTO 550
510 IF N>60 , Q$="O" : GOTO 560
520 Y=1 : X=3+(N-1)*8 : RETURN
530 Y=7 : X=3+(N-16)*8 : RETURN
540 Y=13 : X=3+(N-31)*8 : RETURN
550 Y=19 : X=3+(N-46)*8 : RETURN
560 Y=25 : X=3+(N-61)*8 : RETURN
```


TRS-80: LOAN PAYMENT

BY MILAN CHEPKO

Milan says this program has been tested and approved by the president of his local bank. Dr. Chepko is also writing a program to collect and search data files for some of the bank's customers.

Milan is turning into a one-person computer literacy project in his town of Thief River Falls, MN. His wife, Karen, is now programming and I expect a program or two from his 9-year-old son any day. You will be hearing more from the Chepko Software Consortium–bet on it! – RZ

In searching through the microcomputer literature, I found several loan payment programs, but none of them allow for a so-called "balloon payment" at the end of a specified time interval. This loan is used to keep the monthly payment as low as possible by paying off a lump sum part way through the loan period. As an example, suppose you want to take out a loan on a car, which you will use for 36 months and then sell. Since the car will have some resale value at that time, you could use it to pay off the loan prematurely. In effect, you have taken out a loan for longer than the 36 months to get the benefit of lower monthly payments. You can then terminate the loan when you wish by selling the car and using the car to pay off the remaining principal.

This program seemed difficult to write, until I realized that the original loan could be considered as two separate loans. The first loan is the amount of the balloon payment, or the principal that will be owed after a specified number of months. Each month, you pay only interest on this part of the loan, no principal.

The second loan is the portion that you will be paying both interest and principal on each month. It will be paid off completely by the time you are ready to make the balloon payment.

If this all sounds like getting something for nothing, relax! Since the principal is being paid off slower, you give up some additional interest along the way. Also, you have to hope that the money for the balloon payment will be available when it falls due. (My thanks to Mr. Arnold Carriere for help in explaining this type of loan.)

```
100 CLS:PRINTTAB(8)"*** LOAN PAYMENT PROGRAM ***":PRINT
110 ' MILAN D. CHEPKO  THIEF RIVER FALLS, MN 56701
120 PRINT"SOMETIMES, IT MAY BE DESIRABLE TO ARRANGE A LOAN
130 PRINT"SO THAT THERE IS A SUBSTANTIAL PAYMENT DUE AT THE
140 PRINT"END.  THIS IS CALLED A 'BALLOON PAYMENT' AND COULD
150 PRINT"REPRESENT THE EXPECTED RESALE VALUE OF THE ITEM THE
160 PRINT"LOAN WAS USED TO PURCHASE. ":PRINT
170 PRINT"THIS PROGRAM WILL CALCULATE THE PAYMENTS FOR A LOAN
180 PRINT"WITH OR WITHOUT SUCH A BALLOON PAYMENT.":PRINT
190 L1=0:L2=0:L3=0:I1=0:I2=0:I3=0:P=0:GOSUB460
200 ' *** GATHER NECESSARY DATA ***
210 INPUT"AMOUNT OF THE LOAN=$";L1
220 PRINT:INPUT"% INTEREST/YEAR=";R
230 PRINT:INPUT" #OF MONTHS LOAN WILL RUN=";N
240 PRINT:INPUT"BALLOON PAYMENT AT END=$";L2
250 CLS:PRINT"PAYMENT SCHEDULE FOR LOAN OF $";L1;"OVER THE"
260 PRINT"NEXT ";N; "MONTHS AT ";R; "% INTEREST PER YEAR"
270 R=R/1200:L3=L1-L2:IF L2=0 GOTO 320
280 PRINT:PRINT"BALLOON PAYMENT OF $";L2;"DUE AFTER LAST MONTH
290 '  *** CALCULATE INTEREST ON THE BALLOON  PORTION ***
300 I2=L2*R:I2=(I2*100)+1:I2=(INT(I2))/100
310 '  *** CALCULATE MONTHLY PAYMENT ***
320 X=(1+R)[N:P=L3*((R*X)/(X-1))
330 P=(P*100)+1:P=(INT(P))/100
340 PRINT:PRINT"EACH MONTHLY PAYMENT=$;P+I2
350 PRINT:PRINT"TOTAL PAYMENT BY END OF LOAN=$";N*(P+I2)+L2
360 PRINT:PRINT"TOTAL INTEREST PAID WILL BE $";N*(P+I2)+L2-L1
370 PRINT:PRINT"FOR MONTHLY BREAKDOWN, ";:GOSUB460
380 '  *** GENERATE TABLE SHOWING BREAKDOWN OF PAYMENTS ***
390 I1=0:D=0:GOSUB480
400 FOR C=1 TO N:D=D+1: IF D=13 THEN D=1:GOSUB460:GOSUB480
410 I3=L3*R:I3=(I3*100)+1:I3=(INT(I3))/100
420 I1=I2+I3+I1:Y=P-I3:L3=L3-Y
430 PRINT C,Y,I3+I2,L2+L3
440 NEXT C
450 GOSUB460:GOTO100
460 PRINT"HIT ' ENTER'TO CONTINUE";:INPUT A$
470 CLS:RETURN
480 PRINT"PAYMENT #","PRINCIPAL","INTEREST","BALANCE"
490 RETURN
```

List of variables continued on pg. 57

Making Music On The PET

BY ALFRED J. BRUEY

Now its the PET'S turn to sound off! The software presented here is a simple way to get the idea of making music on the PET. Find out how to hook your PET up to your stereo or build your own amplifier circuit. If you're all thumbs, an easy way out is the commercially available amplifier, called Soundware, from CAP Electronics, Dept. RC, 1884 Shulman Ave., San Jose, CA 95024. – *TD*

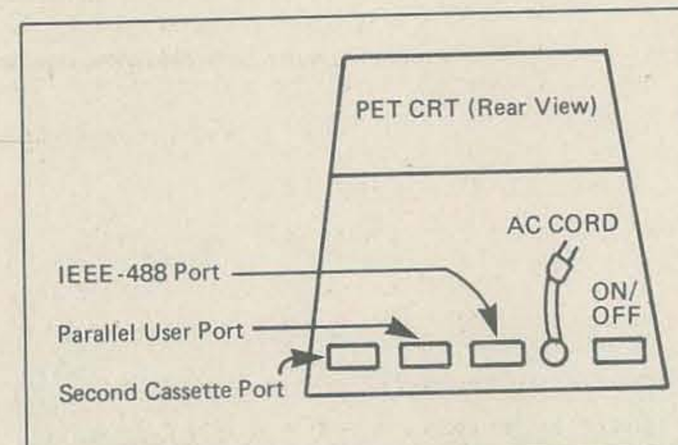
Being able to make music on your PET may not justify its purchase, but it's a lot of fun to do! More PET games are including sound output now and most of them are using the sound generation method described here.

First, we'll take a look at the hardware that you'll need to generate sound. Then we'll cover the software, beginning with a single note and progressing through scale generation, and finally, examine the program necessary to play a complete song.

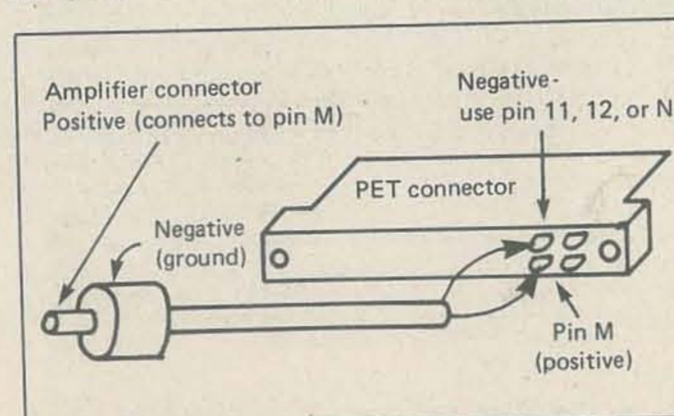
THE HARDWARE

PET music is generated through the CB2 line of the parallel user port of the PET's 6522 VIA (Versatile Interface Adapter). POKEing values into PET locations 59464, 59466 and 59467 sets the frequency and tone of the square waves that are generated by the shift register in the 6522.

The drawing below shows the connections to the PET. Use a connector and don't solder anything to the connectors on the PET. If you are using your hi-fi amplifier for sound output, the connector on the amplifier end (usually an RCA plug type) should match the cable connector.



The edge connector that you need plugs onto the parallel user port of the PET. Do not attach it to the IEEE-488 port. In fact, it's not a bad idea to put a strip of masking tape across the IEEE port so you don't accidentally plug into it. Here's what the completed cable should look like. The amplifier end might look different if your system doesn't use the RCA jack.



You should use shielded cable for the line between the PET and the amplifier. *Don't put the PET connector on upside down!*

First, turn off the PET and the amplifier and then plug the jack into the amplifier. Carefully slide the connector onto the output port of the PET. Be certain that you have the connector on the PET right side up. Now turn on the PET and the amplifier, but set the amplifier volume all the way down to start. The PET puts out a big signal on the CB2 line and you don't want to damage your amplifier.

A word of warning! Avoid discharges of static electricity into your PET. Ground yourself before plugging the connector onto the back of the PET. Be especially careful if you are working in an area that is carpeted with nylon carpeting.

SOFTWARE

Before we proceed with making music, you should know that during some part of your experimentation with music, you may get a musical tone that you didn't expect. No matter what key you hit, it won't stop. To stop the tone without having to turn the PET off and re-enter your program, just enter the command:

```
POKE 59467,0
```

Rather than just giving you a listing of a program and letting you figure it out yourself, let's start with a program to generate one note. Enter and SAVE the following program:

```
10 POKE 59467,16
20 POKE 59466,15
30 POKE 59464,237
40 T = TI
50 IF TI - T<20 THEN 50
60 POKE 59467,0
70 END
```

Now RUN the program and you should hear your first PET music. If you didn't hear a note, check your solder joints and connections. Also make sure that you POKEd the right numbers into the right locations. Then try again. We'll assume everything went right this time and go on with an explanation of the program.

The note that you played was Middle C and it played for 1/3rd of a second. The value 237 in line 30 sets the frequency. It can be any number from 0 to 255. The lower the number, the higher the note. When you try a very low number, like 1 or 2, only your dog can hear it. The value 20 in line 50 tells how long to hold the note in 1/60th of a second units. The 20 holds the note 20/60ths or 1/3rd of a second, a 60 will give you a note 1 second long, a 30 a note 1/2 second long, and so on.

Line 10 starts the shift register in the 6522 to free-running. Line 20 sets the octave. The value of 15 gives a useable range of notes, but you might want to try a 51 or an 85 to see what happens with these values.

The Table of Frequency Constants gives a listing of the numbers to POKE into 59464 to get notes besides Middle C. The formula for calculating the frequency based on values in addresses 59464 and 59466 may also be found there. You now know how to play a note, in fact, you can play any note that is in the range indicated by the equation in the table.

Table of Frequency Constants

Values to POKE into 59464

Note	freq.	POKE	Note	freq.	POKE
B	246.942	251	C	523.25	118
C	261.625	237	C#	554.37	111
C#	277.185	223	D	587.33	104
D	293.665	211	Eb	622.25	98
Eb	311.125	199	E	659.26	93
E	329.63	188	F	698.46	87
F	349.23	177	F#	739.99	82
F#	369.995	167	G	783.99	78
G	391.995	157	Ab	830.61	73
Ab	415.305	148	A	880.00	69
A	440.00	140	Bb	932.33	65
Bb	466.165	132	B	987.77	61
B	493.885	124			

POKEing 59466 with a 51 instead of a 15 will shift all notes one octave higher. POKEing with an 85 will shift all notes two octaves higher.

The value C to be POKEd into location 59464 is given by

$$C = \frac{500000}{fd} - 2$$

where f is the desired frequency, in hz, and $d = \begin{cases} 2 \text{ if } 59466 \text{ is } 85 \\ 4 \text{ if } 59466 \text{ is } 51 \\ 8 \text{ if } 59466 \text{ is } 15 \end{cases}$

Here is a listing to play the C-scale. It will play up and down the scale five times. If you interrupt it in the middle of its song, you might have to POKE a 0 into 59467 as mentioned earlier. Enter and SAVE the following program:

```
10 DATA 0,237,211,188,177,157,140,124,118
20 FOR I = 0 TO 8
30 READ PI(I)
40 NEXT I
42 POKE 59467,16
44 POKE 59466,15
46 FOR CO = 1 TO 5
50 FOR I = 1 TO 8
60 POKE 59464,PI(I)
70 T = TI
80 IF TI - T<6 THEN 80
90 NEXT I
100 FOR I = 1 TO 8
110 POKE 59464,PI(8 - I)
120 T = TI
130 IF TI - T<6 THEN 130
140 NEXT I
150 NEXT CO
160 POKE 59467,0
170 END
```

The DATA statement in line 10 gives the numbers to be POKEd into location 59464 to obtain the various frequencies. Middle C is 237, D is 211, up to 118 for the C above Middle C. Lines 20 to 40 read in the frequency data, line 42 starts the shift register shifting, and line 44 sets up the octave.

Lines 50 to 90 generate the ascending scale and line 100 to 140 the descending one. The length of the notes are set to 1/10th second each by the 6 in lines 80 and 130. The POKEs in lines 60 and 110 set the frequency. The POKE in line 160 stops the shift register. CO is the counter to play the scale five times.

You can now get the PET to play any tune you want. You can set up the proper frequencies in one array and the proper time values for the notes in another, and then play them with one BASIC program. A "Happy Birthday" example is given in lines 10-19 below. Type in, SAVE, and RUN the following program:

```
3 DIM L(25),PI(25)
10 DATA 237,237,211,237,177,188,237,237,211,237
15 DATA 157,177,237,237,117,140,177,188,211
17 DATA 132,132,140,177,157,177
18 DATA 15,5,20,20,20,40,15,5,20,20,20,40
19 DATA 15,5,20,20,20,20,20,15,5,20,20,20,40
20 FOR I = 1 TO 25
30 READ PI(I)
31 NEXT I
32 FOR I = 1 TO 25
33 READ L(I)
40 NEXT I
42 POKE 59467,16
44 POKE 59466,15
50 FOR I = 1 TO 25
63 POKE 59464,PI(I)
70 T = TI
84 IF TI - T<L(I) THEN 84
85 POKE 59464,0
90 NEXT I
100 POKE 59467,0
120 END
```

I'm sure you have the idea by now! You should know that the first three DATA statements contain the frequency values and the next two the corresponding note lengths. Line 84 is the timing loop and the note frequency is set in line 63. The POKE in line 85 is to turn off the sound so that if two consecutive notes are the same, they will sound like two separate notes instead of one long note.

EXTENSIONS

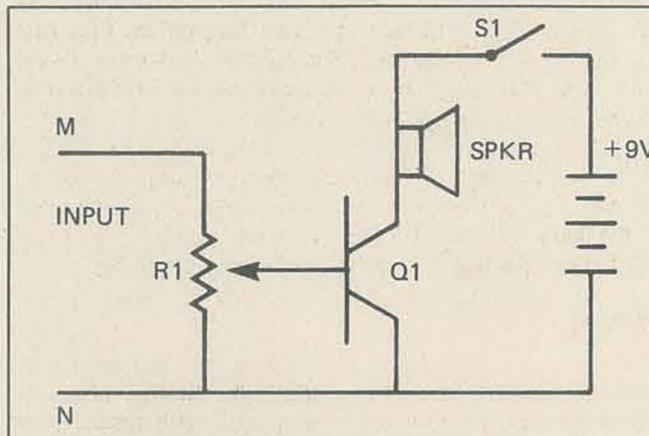
One extension to this music system would be to write a program to convert music symbols, such as C or D#, into their corresponding POKE values. It would be a lot easier to write a C any time you want it than to write a 237.

Another extension would be a program to make your PET keyboard into a musical keyboard. This can be done by taking advantage of the fact that a PEEK on location 515 will tell you what key is depressed. You could also write a program to let the PET compose the music and play it to you. There are two ways to do this: you can either use the RND function to let the PET compose entirely on its own or you can combine the RND function with a set of music-writing rules.

Someday when this music isn't good enough for you anymore, you can start considering different hardware so you can play more than one note at a time. You might also want to consider writing your programs in 6502 machine language instead of BASIC.

A CIRCUIT FOR A PET AMPLIFIER

The following diagram is a circuit for an inexpensive (less than $6.00) PET amplifier for adding music or sound effects to your games.



R1 – 100K potentiometer with switch
S1 – PART OF R1
Q1 – RS2031 (Radio Shack 276-2031)

SPKR – 8ohm Speaker
Small plastic box, 9 volt battery, battery holder

Use an RCA phono jack as the input and you'll be able to use the same connector cable as described previously.

Substitutions may be made for Q1 but for sufficient volume a high-gain transistor should be used. The transistor can be mounted directly on the leads of the potentiometer.

choose-a-title

BY JACK A. TAYLOR
School of Music
Florida State University

For people like myself, who always have difficulty putting titles on stories, articles, poems. . here is an answer from sunny Florida. If I could now get the program to write the paper, I would go to Florida and fish while my microcomputer talks to my editors . . . –RZ

For those college students who struggle to create titles for their research papers, the CHOOSE-A-TITLE system is the solution. This BASIC program uses the RND function* to randomly select five "research type" words from DATA statements 610-820. Line 530 prints the words, appropriately arranging them into a title.

The user can request any number of titles to be printed, and the chances of a repeat are slim. With this program, which was designed for music education students, 673,920 different titles are possible! Of course, one can increase or decrease that potential by adding or deleting research words (and by changing the RND statements).

Dr. Ellis Melton, Coordinator of Music Education at the University of Arkansas, Fayetteville, is the creator of both the CHOOSE-A-TITLE idea and the research words used in this program.

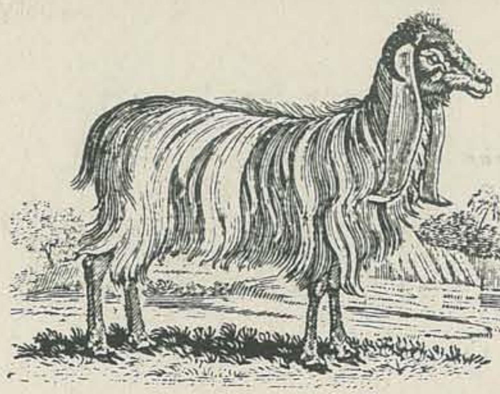
RUN OF THE PROGRAM

```
THIS IS THE MUSIC EDUCATION CHOOSE-A-TITLE
SYSTEM. IT IS DESIGNED TO RANDOMLY SELECT
A TITLE FOR POTENTIAL MUSIC EDUCATION RESEARCH
PAPERS,THESES, AND DISSERTATIONS.


HOW MANY TITLES DO YOU WANT PRINTED ?5

THE DESCRIPTION AND COMPARISON  OF ADDITIONAL  PROBLEMS IN MUSIC APTITUDE

THE OBSERVATION AND CLASSIFICATION OF PRIMARY ALTERNATIVES IN MUSIC FAILURE

THE STUDY AND SCALING OF PRIMARY CONCEPTS IN MUSIC TEACHING

THE MEASUREMENT AND PREDICTION OF CRITICAL ALTERNATIVES IN MUSIC EDUCATION

THE HISTORY  AND APPLICATION OF ADDITIONAL  PATTERNS IN MUSIC
ADMINISTRATION
```

*Format for the RND function will vary, depending upon the BASIC version. The program listed here is Version 2.1, Control Data Corporation (Cyber 70 series computer).

LISTING

```
00010 REM           CHOOSE-A-TITLE SYSTEM
00020 REM
00030 REM           WRITTEN BY JACK A. TAYLOR
00040 REM              TALLAHASSEE, FLA.
00050 REM              JANUARY 12, 1978
00060 REM
00070 PRINT         "THIS IS THE MUSIC EDUCATION CHOOSE-A-TITLE"
00080 PRINT         "SYSTEM. IT IS DESIGNED TO RANDOMLY SELECT"
00090 PRINT         "A TITLE FOR POTENTIAL MUSIC EDUCATION RESEARCH"
00100 PRINT         "PAPERS,THESES, AND DISSERTATIONS."
00110 PRINT
00120 PRINT
00130 PRINT         "HOW MANY TITLES DO YOU WANT PRINTED";
00140 INPUT X
00150 PRINT
00160 DIM A$(20),B$(20),C$(20),D$(20),E$(20)
00170 J=0
00180 REM         STATEMENTS 00220-00510 RANDOMLY SELECT
00190 REM           5 WORDS (VARIABLES A$, B$, C$, D$, E$)
00200 REM           FOR A TITLE FROM DATA STATEMENTS
00210 REM           BEGINNING ON LINE 00610
00220 R1=INT(13*RND(-1)+1)
00230 I=0
00240 READ A$
00250 I=I+1
00260 IF I=R1 THEN 00280
00270 GOTO 00240
00280 R2=INT(15*RND(-1)+1)
00290 I=0
00300 READ B$
00310 I=I+1
00320 IF I=13-R1+R2 THEN 00340
00330 GOTO 00300
00340 R3=INT(16*RND(-1)+1)
00350 I=0
00360 READ C$
00370 I=I+1
00380 IF I=15-R2+R3 THEN 00400
00390 GOTO 00360
00400 R4=INT(12*RND(-1)+1)
00410 I=0
00420 READ D$
00430 I=I+1
00440 IF I=16-R3+R4 THEN 00460
00450 GOTO 00420
00460 R5=INT(18*RND(-1)+1)
00470 I=0
00480 READ E$
00490 I=I+1
00500 IF I=12-R4+R5 THEN 00530
00510 GOTO 00480
00520 REM           PRINT THE TITLE
00530 PRINT "THE ";A$;" AND ";B$;" OF ";C$;" ";D$;" IN MUSIC ";E$
00540 PRINT
00550 REM           STATEMENTS 00570-00590 ALLOW ANOTHER TITLE
00560 REM           TO BE GENERATED AS INDICATED BY INPUT X
00570 J=J+1
00580 RESTORE
00590 IF J<X THEN 00220
00600 REM           VARIABLE A$ DATA
00610 DATA STUDY,INVESTIGATION,SURVEY,DESCRIPTION,HISTORY
00620 DATA MEASUREMENT,ASSESSMENT,OBSERVATION,EVALUATION
00630 DATA DIMENSIONS,RELIABILITY,EFFECTIVENESS,METHODOLOGY
00640 REM           VARIABLE B$ DATA
00650 DATA PREDICTION,ANALYSIS,SCALING,EXPLANATION,COMPARISON
00660 DATA RELATIONSHIP,CORRELATION,PROGNOSIS,CAUSES
00670 DATA APPLICATION,EFFECTS,CLASSIFICATION,REVIEW
00680 DATA VALIDITY,VIABILITY
00690 REM           VARIABLE C$ DATA
00700 DATA SELECTED,PARTICULAR,RANDOM,OVERLOOKED,LITTLE-KNOWN
00710 DATA VITAL,VARIOUS,CERTAIN,PRIMARY,INFORMAL,CRITICAL
00720 DATA THEORETICAL,PRACTICAL,CLASSICAL,ADDITIONAL
00730 DATA GENERALIZED
00740 REM           VARIABLE D$ DATA
00750 DATA FACTORS,TRAITS,HABITS,VARIABLES,CHARACTERISTICS
00760 DATA PROBLEMS,ALTERNATIVES,EFFECTS,PATTERNS,CONCEPTS
00770 DATA THEORIES,MODELS
00780 REM           VARIABLE E$ DATA
00790 DATA APTITUDE,ACHIEVEMENT,STUDY,LEARNING,STUDENTS
00800 DATA TESTS,SUCCESS,FAILURE,ATTITUDES,TEACHING
00810 DATA EDUCATION,CURRICULA,BEHAVIOR,SCHOOLS,HISTORY
00820 DATA ADMINISTRATION,CREATIVITY,ABILITY
00830 END
READY.
```



Newett Awl's Goat Once Again

BY BRIAN YANNY

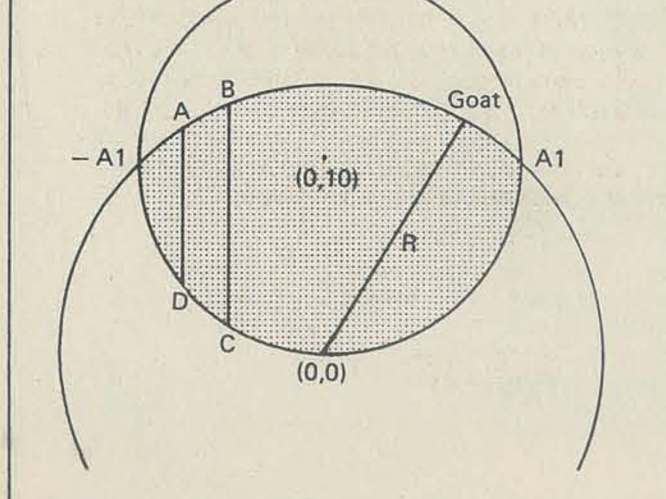
Brian took on the Newett Awl goat problem as a summer school project. The result: a unique way of solving the problem and an A+ in his course (I'm assuming that was his grade). Mr. Kmoch, the instructor, appears to be combining the teaching of good programming style (Brian's listing is well documented) with creative and challenging problems. Summer school will never be dull again! – RZ

The small circle represents the yard. The equation for this circle is $x^2 + (y - 10)^2 = 100$.

The large circle represents the boundary of the range the goat can reach. Its equation is given by $x^2 + y^2 = r^2$ where r is the length of the rope.

By solving the system of the two equations, we arrive at the points – A1 and A1 which will be used as starting and ending values when finding the area of the shaded intersection.

The trapezoid ABCD represents a (not to scale) sample of one of the 50 or so sections the shaded region will be broken into by the program.




```
2!      AUTHOR: BRIAN YANNY
                WASHINGTON HIGH SCHOOL
                MILWAUKEE, WISCONSIN

        COMPUTER: DIGITAL SYSTEMS PDP-11
                  BASIC-PLUS LANGUAGE
                  WITH 4 WORD FLOATING POINT ARITHMETIC
                  (DOUBLE PRECISION)

3!              DESCRIPTION

         THIS PROGRAM COMPUTES THE AREA OF A FIGURE
         BOUNDED BY TWO CIRCULAR ARCS OF DIFFERENT RADII.
         IT DIVIDES THE INTERSECTION OF THE ARCS
         INTO SMALL TRAPEZOIDS, COMPUTES THE AREA OF EACH,
         AND SUMS THE TOTAL.

9!       ITERATIONS
         DETERMINE THE NUMBER OF TRAPEZOIDS THE REGION
         TO BE CALCULATED IS TO BE DIVIDED INTO.

10  INPUT 'ITERATIONS';N
20 IF N<=0 THEN 10 ELSE N=1/N
30 INPUT 'RADIUS OF YARD';Z                 Z is the radius of the yard <10>
35 R=10:M3=1                   R is the beginning value for the length of the rope
                              M3 is the beginning step for increasing rope length
40 DEF FNY(X1)=Z-SQR(Z*Z-X1*X1)     This function is the formula for the upper
                          circle (see diagram) It represents the equation for the yard
41 A=3.14159265358979323*Z*Z/2      A is the desired area the goat is to consume
70 PRINT'WORKING....'
80 PRINT'ROPE LENGTH','AREA GOAT CONSUMES','DESIRED AREA'
90!     START LOOP                                        Set up outside loop
100 T=0     !ZERO THE TOTALLER                       This loop changes rope lengths
110 DEF FNG(X1)=SQR(R*R-X1*X1)     This function is the equation for the lower
                                                       larger circle (see diagram)
115 A1=R*SQR(4*Z*Z-R*R)/(2*Z)      A1 is a variable which is the solution of the
                                                           two circle equations
120 FOR X=-1*A1 TO A1-N STEP N     This for next loop sums and adds together
130T=T+N*(FNG(X)-FNY(X)+FNG(X+N)-FNY(X+N))/2                 the area of each
140 NEXT X                                                   small trapezoid
145 IF T>A THEN 200 ELSE S=T
160 R=R+M3                    This line steps the rope length and repeats the loop
165 GOTO90
200 PRINT NUM1$(R-M3),NUM1$(S),NUM1$(A)      Print partial results in double
                                                              precision format
210 IF ABS(S-T)<1E-9 THEN 500             Check for the end of the program
220 R=R-M3:M3=M3/10:GOTO 90        Otherwise narrow the range by decreasing
500 PRINT:PRINT                                     the step by a factor of ten

510 PRINT 'MAKE THE ROPE 'NUM1$(R)' UNITS LONG' , Here's the answer
520 PRINT
999END



RUN
GOAT6   12:31 PM          31-Jul-79
ITERATIONS? 50
RADIUS OF YARD? 10
WORKING....
ROPE LENGTH   AREA GOAT CONSUMES            DESIRED AREA
11            144.204407260114              157.07963267949
11.5          155.153182398643              157.07963267949
11.58         156.918481699654              157.07963267949
11.587        157.072901601425              157.07963267949
11.5873       157.07951708871               157.07963267949
11.5873       157.07951708871               157.07963267949
11.587305     157.079627345089              157.07963267949
11.5873052    157.079631755343              157.07963267949
11.58730524   157.079632637394              157.07963267949
11.587305241  157.079632659445              157.07963267949
11.5873052419 157.079632679291              157.07963267949
11.5873052419 157.079632679291              157.07963267949

MAKE THE ROPE 11.58730524191 UNITS LONG
```



```
<><><><><><><><><><><><><><><><><><><> <><><><><><><><><><><><><><><><><><><>
<> 1001                        1001 <> <> 1002                        1002 <>
<>                                  <> <>                                  <>
<>                                  <> <>                                  <>
<>            ADMIT ONE             <> <>            ADMIT ONE             <>
<>                                  <> <>                                  <>
<>     THIS IS A GENUINE TICKET     <> <>     THIS IS A GENUINE TICKET     <>
<>                                  <> <>                                  <>
<>           ADULTS $7.50           <> <>           ADULTS $7.50           <>
<>                                  <> <>                                  <>
<>                                  <> <>                                  <>
<> 1001                        1001 <> <> 1002                        1002 <>
<><><><><><><><><><><><><><><><><><><> <><><><><><><><><><><><><><><><><><><>
```

compute-a-ticket

BY DAVID J. BEARD

David's program, written in Programma 2.0 BASIC for a Sphere System Three with a Okidata printer, prints serially numbered tickets, as you like them. David is associated with Sortiarii, a minicomputer hardware and software support firm. His address is RD # 1, Box 648, Newmanstown, PA 17073.

If you use this program to generate tickets for any good events, make sure you send David a set of "comps." – RZ

```
10  REM -----"TICKET PRINTING PROGRAM"
20  REM -----"79/08/08 DJB"
30  REM -----"THIS PROGRAM USES 8.5 BY 24 INCH PAPER"
40  REM -----"IT PRINTS 16 NUMBERED TICKETS PER SHEET"
50  DIM A$(38),B$(38),C$(38),D$(38),E$(38),F$(38),G$(38),H$(38)
60  LET A$="<><><><><><><><><><><><><><><><><><><>"
70  LET B$="<>                                  <>"
80  LET C$="<>            ADMIT ONE             <>"
90  LET D$="<>     THIS IS A GENUINE TICKET     <>"
100  LET E$="<>           ADULTS $7.50          <>"
110  CLEAR   \ PRINT   \ PRINT   \ PRINT
120  PRINT "--PREPARE THE PRINTER BEFORE        PROCEDING"
130  PRINT \ PRINT \ PRINT
140  PRINT "--FIRST TICKET NUMBER?";
150  INPUT T
160  CLEAR
170  FOR L=0 TO 7
180  LET T1=T+2*L\T2=T+2*L+1
190  PRINT #1A$;A$
200  GOSUB 350
210  PRINT #1B$;B$
220  PRINT #1B$;B$
230  PRINT #1C$;C$
240  PRINT #1B$;B$
250  PRINT #1D$;D$
260  PRINT #1B$;B$
270  PRINT #1E$;E$
280  PRINT #1B$;B$
290  PRINT #1B$;B$
300  GOSUB 350
310  PRINT #1A$;A$
320  PRINT #1
330  NEXT L
340  GOTO 60
350  REM ----- SERIAL NUMBERS
360  PRINT #1"<>";T1;"                        ";T1;"<>";
370  PRINT #1"<>";T2;"                        ";T2;"<>"
380  RETURN
```

An Introduction to Recreational Disputing

BY JOHN HIRSCH

At times, the pressures of our fast-paced technological society begin to affect each of us. To relieve the tensions, some of us jog, some play tennis, and some beat out their exasperations on the typewriter. John chose the last outlet. Here is the result of his exercise in typewriter therapy . . . – *RZ*

What's the difference between a subprogram, a subroutine, and a library function? What is the distinction between a real-time system, an on-line system and a time-sharing system? What distinguishes a programmable microprocessor from a microprogrammable processor?

If you can't answer each of the questions above in 25 words or less, don't let it worry you. The invention of new jargon is more often a Humpty-Dumpty procedure where words mean what I say they mean and you are free to knock me off the wall if you can. Each computer manufacturer has their own in-house glossary and usually feels free to pluck any word out of the common stock and reshape it to fit their purposes.

Aside from this problem of ad hoc definition, there is also the great gray area that keeps creeping up on us. We start with a cow and a horse, which are easy enough to tell apart even if you lack a degree in animal husbandry. Then someone comes along breeding either mares that give milk with a high butterfat content or galloping bovines intended for the Belmont Stakes, then the great gray area arrives.

There used to be large computers that were enthroned with all their satellites behind glass walls, friendly minicomputers for desktop use, and microcomputers with a central processor sandwiched onto a single chip. Now minicomputers are growing larger and possess their own train of peripherals supervised by an elaborate operating system. Large scale integrated circuits have been applied to the processors of all computers, and bit-slicing techniques can be used to alter computer word sizes. Biological beings seem to be governed by some natural laws that have remained immutable up to the present. But electronic husbandry allows any sort of mating in endless permutation. It is inevitable that the distinction between species is rapidly blurred.

Now it is even difficult to find an adequate definition for "computer." The word is being applied to any dinky microprocessor used to control an eggbeater or an electric toothbrush. There are a plethora of dictionaries which attempt to organize the chaos of computer terminology. They trap each feral concept and cram it into its own neat cage. But concepts rarely remain domesticated when you apply them to familiar computer systems and languages.

A subroutine in FORTRAN, for instance, is equivalent to a subprogram in COBOL, but not to a subroutine in BASIC, which does the same thing as a "perform statement" in COBOL.

Now that we have dealt with subprograms and subroutines, we can go on to library functions, which are like subroutines except that they must be invoked with at least one argument to which they apply their function and return an answer. The library of library functions may be a collection of useful subroutines written by a software supplier, with a directory that the computer knows how to access. However, in some assembly languages library functions are user-defined but have a linkage method that differs from ordinary subroutines. No need for any further discussion of linkage methods because everyone knows about *them*.

If all this seems perfectly clear, let's go on to answer the other questions posed at the beginning of this article. A time-sharing system is one in which a fast sequential processor fools all its customers into thinking that they are all being served at once. It does this by working on each program for only a fraction of a second. Then the processor goes on to the next user in either a round-robin sequence or some order specified by the system designer.

On-line systems make use of time-sharing to service business customers, as in the terminals of an airline reservation system, for instance. Real-time systems also use a time-sharing concept. However, the term "real-time" is usually applied to process control where a central processor supplies the intelligence for any number of subsidiary machines and must send and receive messages at high speed. A strict definition of a real-time system requires that incoming data be processed and used in a program while the program is actually running.

The difference between a programmable microprocessor and a microprogrammable processor is an easy one. All microprocessors with access to read-write memory are programmable. Most processors of any size have circuits to directly execute their primitive machine language. A few large computers and some minis have hardware that is left in a more primitive state. They require a program in ROM (read-only memory) in order to execute their machine language. This is a microprogram and is written in a more primitive instruction set than machine language. Microprogrammable processors are slow, but flexible. There are microprogrammable microprocessors around, although no one has yet incorporated one into a home computer.

By now, you will have noted that the definition of all but the simplest of computer terms requires the use of other equally obscure terms. In addition, authors with the courage (or naivete) to compile glossaries of computer terms may find that all their efforts to clarify only add to the confusion. It is possible that every published definition gives birth to two or more terms by way of clarification. Jargon is increasing at some geometric rate while our brains have arithmetic limits.

To help solve this dilemma, the Recreational Disputing Society (RDS) has been formed. There are no dues or organized activities. Members are easily recognized by their ability to draw hair-line distinctions among computer terms that are seemingly equivalent. This computer activity is absorbing and endlessly inventive and it *doesn't* require a computer.

The RDS does not issue membership cards. Members are born, not made. The *only* rule in the by-laws covers disputes between two members that are not settled by an ultimate meeting of minds. In these cases, an appeal can be made to a third RDS member. The by-law states that the third member is required to give several cogent reasons why both the other two people are wrong!

Continued from pg. 47

LIST OF VARIABLES USED

- L1. . . initial loan
- L2. . . balloon payment
- L3. . . rest of loan, on which principal will be paid monthly
- I1 . . . total interest to date
- I2 . . . monthly interest on balloon principal
- I3 . . . monthly interest on rest of loan
- P . . . monthly payment on rest of loan (interest + principal)
- P+I2 . total monthly payment
- R . . . interest rate (initially %/year, later converted to %/month)
- N . . . number of months until balloon payment due

Note: the [in line 320 is . . . this program would probably work in Level I by writing subroutine to handle the exponent.

Continued from pg. 22

Cryptarithms
Puzzle 14 (Extra Clues)

E * E = T; so T is 9, 4, or perhaps 5 (with a carry). Thus E can be either 2 or 3. In the first product, R * U = E, so E cannot be 3. Why? So E is 2. At this point, begin using an "Is-Not" Chart. Now by looking at the ninth column where R + T + W = W you can begin to unravel the rest. The Elimination Table or "Is-Not" Chart will help you with this task. Good luck!

Reviews

MICROCOMPUTERS AND THE 3Rs, A GUIDE FOR TEACHERS
By Christine Doerr
Hayden Book Company, Inc.
Rochelle Park, N.J.
$7.95

I'm quite pleased to have had the opportunity to review this book. It fills the need for a practical guide for teachers who want to get involved in computing. The comprehensive introduction to educational computing and the reliable suggestions on selecting a unit should be of value to experienced instructional computer users and administrators as well. The treatment of important questions about using the computer in teaching also makes it recommended reading for any teacher or administrator getting involved for the first time, whether they are considering microcomputers or time-sharing systems.

The highlights of the book are as follows: In "The Microcomputer: What it is and What it Ain't," the author explores the history of instructional computing and its status in schools today. She also discusses the components of a microcomputer system and gives some advice on economic choices one may face when selecting equipment.

"What the Micro Computer Offers to Education" addresses common fears and concerns regarding the computer in education, and the question of "why get involved?" The author suggests that the answer is the opportunity for great teaching. The development of that theme offers sensible priorities.

For the teacher deciding whether to build or buy a micro-system, "How to Get Started" provides plenty of information. (Building a unit is not recommended for most applications.) Additionally, the author examines important logistical problems such as access to the machine and necessary security.

The chapters on computer science, problem solving, instructional simulation and games offer information that's fairly well known to old hands, but provide a concise, comprehensive introduction to newcomers. The material selected is good, and as is the case throughout the book, a copious list of references is provided.

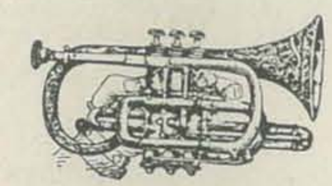
The final two chapters offer advice on Computer Assisted Instruction (CAI) and Administrative Computing, two topics which are the source of considerable confusion to educators. The author makes thorough observations on the pros and cons of using CAI, differentiates among the basic types, and provides some simple examples.

A few words of caution are given about using micros for administrative computing. I personally would be more conservative; at present, most major administrative applications do not belong on a micro. A great deal more than writing or procuring programs is required. The whole man-machine system must be carefully designed and tested, otherwise the potential for failure and the resulting frustration is great.

My criticism is minor. The author's personal choice of a computer and some of the general recommendations accompanying it are a bit dated. For instance, a floppy disc was not included on her system and not recommended on a starter system because of cost. Floppy discs are now available for some systems for as little as $600, and their availability can make an enormous improvement in its usefulness to an educational program. In fairness, this recommendation was probably made with the best information available at the time, but things are changing very rapidly in this field. The author compensates adequately by comparing the most popular microcomputer systems and avoiding specific recommendations on choice of a particular piece of equipment.

It is on the whole a good effort. Chris Doerr has put her years of exposure to education and computing to good use with this comprehensive introduction to instructional computing. She shows, in addition, awareness of the financial realities of school systems. Because of the invaluable and unbiased information it contains, the book deserves wide acceptance.

Reviewed by John Nierengarten
University of Wisconsin
La Crosse, Wisconsin 54601



THE INCREDIBLE SECRET MONEY MACHINE
By Don Lancaster
Howard W. Sams & Co., Inc.
Indianapolis, Indiana
$5.95

Before you read this book you may want to quit your job, if you've got just *one*, because, according to the author, "(It) will certainly be using you to its own profit, having you do what it wants . . . and may even insist on such dumb things as your being at a certain place at a certain time, wearing specified costumes or paying tribute to others in its hierarchy." After you've quit, or *in order to*, settle down to read this engaging book of "contrary opinions."

Mr. Lancaster is known for his earlier books *CMOS Cookbook*, and *Cheap Video Cookbook*. In *The Incredible Secret Money Machine*, Lancaster draws upon his own experience and that of other people to create this practical guide for those with the urge for financial independence. The author talks taxes and copyrights, and proposes some novel investment ideas. Actually, this is a book you'll learn from even if you won't be starting your own business. Numerous examples and references to computer-related business should be of special interest to *RC* readers.

You'll really be excited by this book if you're itching for a job change and have a business idea. Contrarily and with much humor, Lancaster urges you to eliminate such words as *salesman*, *agent*, and *secretary* from your vocabulary, and to forget about much-touted techniques like *creative management* and *aggressive marketing*. Instead, he advises you to employ exactly 0.834 employees (that's 83.4% of your time) and to do everything yourself. His most binding prerequisite is that you *want* to and *will* spend most of your time working on your project.

Sound intriguing? It's only the beginning. Lancaster advises nitty gritty measures that will help you avoid pitfalls and achieve success. "If a person seeks you out, he has made an ego decision that he will defend. Should he buy from you, he will defend this decision . . . opposite-ly, if a person feels he was 'sold' some-thing, he will easily find minor and even unreasonable faults with it . . .

". . . Like to travel? Do stories or photos on where you go and let books, magazines and papers pay at least part of your way." You'll learn why 'grungy' places are the best places to rent, and why word of mouth will be your best kind of advertis-ing. You'll learn by diagram a method of modifying your television to help you with your secret money machine. It's not fair of me to tease you like that. This is one of the many illustrations throughout the book, and could be titled 'Four Steps to Disconnect.'

Learning to communicate in both words and images is a big part of becoming a success on your own, Lancaster believes. He covers resources to use for writing, the basics of graphic production for publication. For example, "if you do too good a job . . . it may even hurt you . . . the editor can use it directly, rather than have an artist redo it."

Finally, you'll learn about "unmatters." There must be one honcho, and not two, in your secret money machine. If there are two involved in your project, one must be a honchee!

In two days my review copy of this book was marked up by three friends. Nuff said!

Reviewed by Sara Werry
People's Computer Company
Menlo Park, CA 94025

U.S. POSTAL SERVICE
STATEMENT OF OWNERSHIP, MANAGEMENT AND CIRCULATION
(Required by 39 U.S.C. 3685)

1. TITLE OF PUBLICATION: RECREATIONAL COMPUTING — A. PUBLICATION NO.: 3 5 6 8 7 0 — 2. DATE OF FILING: 28 Sep 1979

3. FREQUENCY OF ISSUE: bimonthly — A. NO. OF ISSUES PUBLISHED ANNUALLY: six — B. ANNUAL SUBSCRIPTION PRICE: $10.00

4. LOCATION OF KNOWN OFFICE OF PUBLICATION *(Street, City, County, State and ZIP Code) (Not printers)*: 1263 El Camino Real, Menlo Park, CA 94025 (San Mateo County)

5. LOCATION OF THE HEADQUARTERS OR GENERAL BUSINESS OFFICES OF THE PUBLISHERS *(Not printers)*: same

6. NAMES AND COMPLETE ADDRESSES OF PUBLISHER, EDITOR, AND MANAGING EDITOR

PUBLISHER *(Name and Address)*: Willard J. Holden, 1263 El Camino Real, Menlo Park, CA 94025

EDITOR *(Name and Address)*: Bob Albrecht/Tracy Deliman/Ramon Zamora (same)

MANAGING EDITOR *(Name and Address)*: same as editors

7. OWNER *(If owned by a corporation, its name and address must be stated and also immediately thereunder the names and addresses of stockholders owning or holding 1 percent or more of total amount of stock. If not owned by a corporation, the names and addresses of the individual owners must be given. If owned by a partnership or other unincorporated firm, its name and address, as well as that of each individual must be given.)*

NAME	ADDRESS
People's Computer Co.,	1263 El Camino Real, Menlo Park, CA 94025

8. KNOWN BONDHOLDERS, MORTGAGEES, AND OTHER SECURITY HOLDERS OWNING OR HOLDING 1 PERCENT OR MORE OF TOTAL AMOUNT OF BONDS, MORTGAGES OR OTHER SECURITIES *(If there are none, so state)*

NAME	ADDRESS
none	

9. FOR COMPLETION BY NONPROFIT ORGANIZATIONS AUTHORIZED TO MAIL AT SPECIAL RATES *(Section 132.122, PSM)*
The purpose, function, and nonprofit status of this organization and the exempt status for Federal income tax purposes *(Check one)*

[XX] HAVE NOT CHANGED DURING PRECEDING 12 MONTHS [ ] HAVE CHANGED DURING PRECEDING 12 MONTHS *(If changed, publisher must submit explanation of change with this statement.)*

10. EXTENT AND NATURE OF CIRCULATION	AVERAGE NO. COPIES EACH ISSUE DURING PRECEDING 12 MONTHS	ACTUAL NO. COPIES OF SINGLE ISSUE PUBLISHED NEAREST TO FILING DATE
A. TOTAL NO. COPIES PRINTED *(Net Press Run)*	12,300	9,200
B. PAID CIRCULATION 1. SALES THROUGH DEALERS AND CARRIERS, STREET VENDORS AND COUNTER SALES	3,359	2,854
2. MAIL SUBSCRIPTIONS	5,097	4,975
C. TOTAL PAID CIRCULATION *(Sum of 10B1 and 10B2)*	8,456	7,829
D. FREE DISTRIBUTION BY MAIL, CARRIER OR OTHER MEANS SAMPLES, COMPLIMENTARY, AND OTHER FREE COPIES	1,195	300
E. TOTAL DISTRIBUTION *(Sum of C and D)*	9,651	8,129
F. COPIES NOT DISTRIBUTED 1. OFFICE USE, LEFT OVER, UNACCOUNTED, SPOILED AFTER PRINTING	2,270	1,035
2. RETURNS FROM NEWS AGENTS	379	36
G. TOTAL *(Sum of E, F1 and 2—should equal net press run shown in A)*	12,300	9,200

11. I certify that the statements made by me above are correct and complete. — SIGNATURE AND TITLE OF EDITOR, PUBLISHER, BUSINESS MANAGER, OR OWNER: Willard J. Holden

12. FOR COMPLETION BY PUBLISHERS MAILING AT THE REGULAR RATES *(Section 132.121, Postal Service Manual)*

39 U. S. C. 3626 provides in pertinent part: "No person who would have been entitled to mail matter under former section 4359 of this title shall mail such matter at the rates provided under this subsection unless he files annually with the Postal Service a written request for permission to mail matter at such rates."

In accordance with the provisions of this statute, I hereby request permission to mail the publication named in Item 1 at the phased postage rates presently authorized by 39 U. S. C. 3626.

SIGNATURE AND TITLE OF EDITOR, PUBLISHER, BUSINESS MANAGER, OR OWNER: Willard J. Holden PUBLISHER

PS Form 3526 *(Page 1)*

Announcements

Hardware

SuperTalker. This new peripheral system allows the Apple II computer to output exceptionally high quality human speech through a loudspeaker under program control. Output may also be directed through any P.A. or stereo system. It consists of the peripheral card, a microphone, a loudspeaker, software and documentation and two SuperTalker programs. $279 assembled and tested. Contact: Mountain Hardware, Inc. 300 Harvey West Blvd., Santa Cruz, CA 95060 (408) 429-8600.

Malibu 165 Printer. This new model can be operated in three modes: as a high-speed dot matrix printer at 165 characters per second; a reduced speed, letter-quality dot matrix printer at 90 cps; or a full-graphics matrix printer. Many new features. Price is $2395.00. Deliver 30 days ARO. For more information contact Malibu Design Group, Inc., 8900 Eton Ave., Suite G, Canoga Park, CA 91304, (213) 998-7694.

Apple Computer I/O Card for Malibu 165 Printer. This interface card allows the Apple computer to interface directly to the basic Malibu Model 165 dot matrix printer, taking advantage of the unique graphics capabilities of both the Apple and the Malibu printer. $260.00 with cable and instruction manual. For more information contact Malibu Design Group Inc., 8900-G Eton Ave., Canoga Park, CA 91304, (213) 998-7694.

M^2-250 Dual Disk Drive. This Micro Squared unit is capable of single or double density and consists of two double-sided drives, a power supply, cable, and chassis and sells for $1195. It has 140 track capacity, with double density feature allowing 875K bytes of storage, is completely assembled and immediately available. Contact Micro Squared Inc., Suite 5B, 5131 Owensmouth Ave., Canoga Park, CA 91303, or call (213) 883-1993.

Computeacher™. A unique self-study training system designed to teach personal home computing at a low cost in a minimal amount of time is now available from Sybex. The package includes a microcomputer board, program cassette, two educational books (*Programming the 6502* and *6502 Applications Book*), required manuals and instructional audio casette. Retails for $299. For more information contact Sybex, 2020 Milvia St., Berkeley, CA 94704. (415) 848-8233.

Disk Drive Head Cleaner. This is for the TRS-80. It is a mini diskette, reusable on both sides. Includes a program that cleans thoroughly and automatically, and cleaning solution. Head cleaner will allow more reliable disk drive operation. $12.95 from dealer or The Bottom Shelf, Inc., P.O. Box 49104, Atlanta, Georgia 30359. (404) 939-6031.

Computer and Terminal Unit. This functions as a full duplex, dumb terminal which can be used in conjunction with remote time-sharing services, and also as an advanced personal computer. It combines PET computer and NCE's own design, using BASIC (level II). It is available with 8K, 16K of RAM memory, and 14K ROM memory. Peripherals available. Introductory price is $795. NCE/CompuMart, PO Box 8610, Dept. P1, Ann Arbor, Mi. 48107. (313) 994-3200.

Software

Basic Compiler. An efficient compiler for 8080 and Z80 CP/M systems that supports extensive, commercial features of Microsoft BASIC-80. Produces optimized, relocatable machine code in Microsoft's standard binary format; compiled BASIC programs can be loaded and linked with subroutines generated by Microsoft's FORTRAN-80 and COBOL-80 compilers, and MACRO-80 assembler. Single copy $395. For more information contact Microsoft, 10800 NE 8th, Suite 819, Bellevue, WA 98004 (206) 455-8080.

Fantasy Adventure Game. Pacifica™ is an action-packed game for the Apple computer. Set sail from the ancient port of No Return, cross the uncharted Sea of Blood, brave the dangers, and you will find yourself on the mysterious island of Pacifica. For a copy send $9.95 to Rainbow Computing, 17023 White Oak Ave., Granada Hills, CA 91344.

Music Program. Software Music Synthesis System is a software/hardware package that turns any 8080/Z-80 or 8085 microcomputer into a high quality, multi-voiced music synthesizer. Software includes line oriented text editor, high level music language compiler, file management and music synthesizer. The price with documentation, hardware and ten songs is $79.95. Those interested contact California Software, P.O. Box 275, El Cerrito, CA 94530 (415) 527-7730.

Small Business Programs. CALC is a program that can save a lot of time and money. It selects bank accounts—enters checks, calculates deposits and displays grand total. Other programs in package print labels and provide sophisticated check balancing. For TRS-80 level II and 16K memory. For more information, Micro Architect, 96 Dothan St., Arlington, MA 02174.

Checkbook II. This program does everything necessary to keep checkbook balanced and then some. It can handle amounts up to $1,000,000 using codes of up to four alpha or numeric characters, and it does a complete balance, reconcile, and editing. For the TRS-80 with 16K or more. Price is $18.50. For more information contact The Bottom Shelf, Inc., Box 49104, Atlanta, Georgia 30359. (404) 939-6031.

Telephone Dialer Program. The Z80 program allows the TRS-80 Level II to dial the telephone. The program can aid the handicapped or assist the baby sitter. Twenty phone numbers may be dialed, accessed using the letters A through T. Both dial and push-button phones may be used. $7.95 includes program, instructions, interface circuit diagram and parts list. Write: Software Exchange, 2681 Peterboro, W. Bloomfield, Mi. 48033.

Three Mile Island. Could it have been prevented? Decide for yourself. Muse Co. has an Apple II disk simulation that lets you run the reactor. The cost is $39.95, and the program requires 48K of memory. Muse Co. Also has several other interesting programs on tape and disk. Write to: 7112 Darlington Dr., Baltimore, MD 21234.

Programming Tools. Infinite Basic is a revolutionary new product adding over *70* BASIC non-trivial commands to TRS BASIC. Combinations of these added BASIC commands can be packaged and loaded into selected memory locations to minimize memory requirements, and to provide programmers with the sophisticated tools necessary to develop professional packages. Available for $49.95 from RACET, 702 Palmdale, Orange, CA 92665, (714) 637-5016 and from dealers.

Educational Software

Skills and IQ builder. This is a series for PET, Apple II and TRS-80 level 1 or level 2. More than two dozen new programs offered for preschool, elementary, junior high, high school and adult levels. Includes games, tutorials, practice, memory, spelling and vocabulary builders, crosswords and other innovative programs. Prices range from $13.50 to $18.50. For further information contact Program Design, Inc. 11 Idar Court, Greenwich, Conn. 06830, (203) 661-8799.

Teacher For the TRS-80. An effective program for computer-assisted instruction using the TRS-80 16K Level II. TEACHER will structure and present tests of up to 20 questions on virtually any subject. Uses technique of positive and immediate reinforcement. Sells for $9.95, with an instruction booklet and blank data cassette. Contact area computer stores or Instant Software, Inc., Peterborough, NH 03458, (603) 924-7296.

Individual Study Center. This is the newest educational course in the Teach-Yourself-By-Computer series. It is designed to teach any subject in a fun and challenging manner, for a variety of activities. There are many prepared subject data tapes for grades 1-8, high school through adult. Activities include Puzzler, Beat the Clock, Matching and Completion Drills. For the TRS-80 Level II, 16K, cost is $39.94 for package. For a brochure and tapes contact SOFTWARE™, 40 Stuyvesant Manor, Geneseo, New York 14454, (716) 243-3005.

Publications

PET Gazette. A new name for the Gazette: *Compute*, the journal for progressive computing. Robert Lock, President of Small Systems Services, Inc. of Greensboro, NC and Len Lindsay, former publisher of the PET Gazette, announced the acquisition of the PET Gazette by Small Systems Services, Inc. They are soliciting articles for future issues, and plan to begin bi-monthly issues with the January/February issue in 1980. The new address for Advertising, Subscriptions, Dealer Information and Article Submission is COMPUTE, The Journal for Progressive Computing, P.O. Box 5119, Greensboro, NC 27403. Telephone (919) 272-4867. Editorial questions should be addressed to Mr. Len Lindsay, 1929 Northport Dr. #6, Madison WI 53704.

More Basic Computer Games. a sequel to *Basic Computer Games* by David Ahl. This book contains 84 new games. It includes games such as Camel, Father, Maneuvers, Close Encounters and others. Each game is complete with description, program listing and sample run. The hilarious drawings by George Beker make the book a visual delight too. $7.50 from Creative Computing Press, P.O. Box 789-M Morristown, NJ 07960 (201) 540-0445.

Mass Storage Systems for the TRS-80. This booklet outlines the various methods that the TRS-80 user can use to load and store his programs off-line. It describes cassette systems, mini-floppy disk systems, full-size floppy disk systems, data cartridge, high speed cassette, and proposed hard disk systems. Includes other features. No charge from Parasitic Engineering, Box 6314, Albany, CA 94706 (415) 527-6133.

Systems Extensions. This 128 page journal consists of one half text material and one half catalog. It gives a general overview of data processing with particular points of interest for TRS-80 owners, and provides a source of supply of over three hundred useful items for micros. $3.00 from The Bottom Shelf, Inc., P.O. Box 49104, Atlanta, Georgia 30359. (404) 939-6031.

TRS Yellow Pages. Issue 1.4, is a twelve-page publication devoted entirely to business software. It is a handy guide for selecting business software for the TRS-80. It describes all the software produced by Micro Architect. This issue features a sophisticated data base manager for the TRS-80. For free copy, send two stamped, long, self-addressed envelopes to Micro Architect, 96 Dothan St., Arlington, MA 02174.

Renaissance Man: The Key Component. Over 360 pages of presentations from the 1979 AEDS Convention held May 15-18, 1979, Detroit, Michigan. Topics include: computer-assisted instruction, instruction learning applications, computer-related curriculum, education administrative applications, computer resources, data-center administration. Send $12.00 to AEDS Proceedings, 1201 16th St., N. W., Washington, D.C. 20036.

Programming the Z80. This informative introductory text on programming by Rodnay Zaks offers a comprehensive description of the Z80 instruction set and a thorough account of its internal operations. Includes an extensive chapter on data structures with lists, tables, binary trees, even hashing and other algorithms. For more information contact Sybex, 2020 Milvia St., Berkeley, CA 94704 (415) 848-8233.

Other

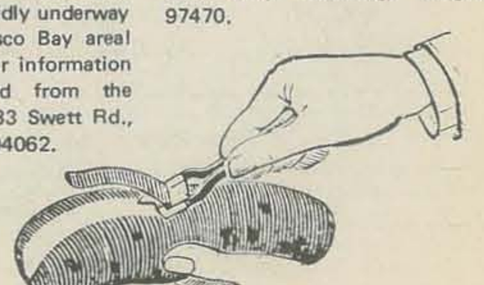
Computer Faire. The 5th West Coast Computer Faire has been scheduled to take place in San Francisco's Civic Auditorium & Brooks Hall, March 14-16, 1980, Friday through Sunday. This is a change from an original proposal that the 5th Faire be held in Los Angeles next November—a proposal that was canceled some months ago. This is also a change from a more recently announced date in San Francisco, scheduling the 5th Faire for the end of February and first of March. This final change of dates was made because the Civic Center just became available for March 14-16, and the Faire organizers felt it was a preferable time—the weather would be better and Spring would be solidly underway in the San Francisco Bay area! Exhibitor & speaker information may be requested from the Computer Faire, 333 Swett Rd., Woodside, CA 94062. (415) 851-7075.

Consumer Computer Information Service. Shoppers for computers in the Southern California area will now find their task greatly simplified, thanks to a new service implemented by the Southern California Computer Dealers Association. Everyone from San Diego to Santa Barbara may dial the toll free number: (800) 432-7257. The caller will be given names, addresses and phone numbers of the three computer stores nearest him.

Independent News Line. Users of personal computers made by the Heath Company may now obtain news bulletins of interest to them by calling a telephone number in Washington, DC. Recorded announcements are provided for subscribers to *Buss: The Independent Newsletter of Heath Co. Computers*. No modem is required to use the *Buss* system. Further information is available from Buss, 325-J Pennsylvania Ave., S.E., Washington, DC 20003.

Users' Group. DC Area Apples: Washington Apple Pi meets 9:30 AM, the fourth Saturday of each month at George Washington University, Room 206, Tompkins Hall, 23rd & H Streets N.W., Washington. For further information, write Washington Apple Pi, PO Box 34511, Washington, DC 20034 or call Sandy Greenfarb (301) 674-5982.

Programming Contests. Dr. David Moursund of the University of Oregon and his task force have put together materials to aid high schools, colleges, or individuals in planning high school programming contests. The material can be used for team contests or individual competitions. Packet includes rules, sample problems, references, and solutions. Request information from: Dale Bryson, Mathematics Department, Umpqua Community College, P.O. Box 967, Roseburg, Oregon 97470.



AMINAL A Computer Game

BY KATHY BURK, AGE 10 &
RACHEL WASSERMAN, AGE 9

Part I: The Challenge

Here is a sample run of a program designed by these two young ladies. The challenge to the readers: What does the program look like? We will publish their program next issue plus the responses we get. Oh, yes!! I almost forgot. You can't send us a solution to this puzzle unless you are less than 10 years old. Just include a note from an adult verifying your age or a copy of your birth certificate. — RZ

The way we made this program was by using imagination, a computer and hard work.

The reason why we made this program is for fun, because we wanted to learn about computers, and because we wanted to make our own program.

Sample Run:

```
TYPE YOUR NAME, PLEASE
?GERTRUDE
IN THIS GAME YOU MAKE YOUR OWN ANIMAL, GERTRUDE
WHAT DO YOU WANT YOUR ANIMAL TO BE, GERTRUDE
?HIPPO
WHAT SIZE DO YOU WANT YOUR ANIMAL TO BE, GERTRUDE
?HUMUNGUS
NOW YOU HAVE A HUMUNGUS HIPPO, GERTRUDE
WHAT COLOR DO YOU WANT IT TO BE, GERTRUDE
?PINKISH
WHAT COLOR EYES DO YOU WANT IT TO HAVE
?PURPLE
NOW YOU HAVE A HUMUNGUS PINKISH HIPPO WITH PURPLE EYES
HOW MANY LEGS DO YOU WANT IT TO HAVE, GERTRUDE
?NINETEEN
NOW YOU HAVE A HUMUNGUS PINKISH HIPPO WITH NINETEEN
LEG(S) AND PURPLE EYES
WHERE DO YOU WANT IT TO LIVE
?JUNGLE
NOW YOU HAVE A HUMUNGUS PINKISH HIPPO WITH NINETEEN
LEG(S) AND PURPLE EYES THAT LIVES IN A JUNGLE
WELL, BYE, GERTRUDE
```