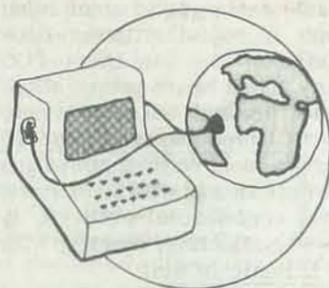


Second Class Postage Paid at Menlo Park, CA
Address Correction Requested

93065 JOHN SB09 01 2251
BYRON JOHNSON MY81
356 LAGUNA TERR
SMT VALLEY, CA 93065



Dr. Dobb's Journal is not only a highly respected reference journal, but a lively forum for the more advanced home computerist.

DR. DOBB'S JOURNAL of COMPUTER Calisthenics & Orthodontia *Running Light Without Overbyte*

COMPLETE SYSTEMS & APPLICATIONS SOFTWARE

User documentation, internal specifications, annotated source code. In the three years of publication, *DDJ* has carried a large variety of interpreters, editors, debuggers, monitors, graphics games software, floating point routines and software design articles. Recent issues have highlighted:

- An Interactive Timeshared 8080 Operating System
- Tiny Grafix for Tiny Basic
- The Heath H-8 System
- A KIM/6502 Line Editor
- Lisp for the 6800
- Dumping Northstar Disk Files
- A 1K Utilities Package for the Z80

OUR READERS SAY

"You maintain a high quality in your *Journal* that fills a gap in the microprocessor field."

"*Dr. Dobb's Journal* will soon be recognized as a true scientific journal in the heartland of micro software."

"One of the reasons I subscribe to *DDJ* is that it is the best source of articles on doing big things on small computers."

"*Dr. Dobb's Journal* remains uniquely free from hype; continues to be intelligent, critical, stimulating—don't let any of this change."



People's Computer Company 1263 El Camino Real, Box E, Menlo Park, California 94025

Dr. Dobb's Journal is published 10 times a year by People's Computer Company, a non-profit educational corporation. For a one-year subscription, send \$15 to *Dr. Dobb's Journal*, Department C3, 1263 El Camino Real, Box E, Menlo Park, CA 94025 or send in the postage-free card at the center of this magazine.

FORMERLY PEOPLE'S COMPUTERS

In Canada \$2.50 \$2.00

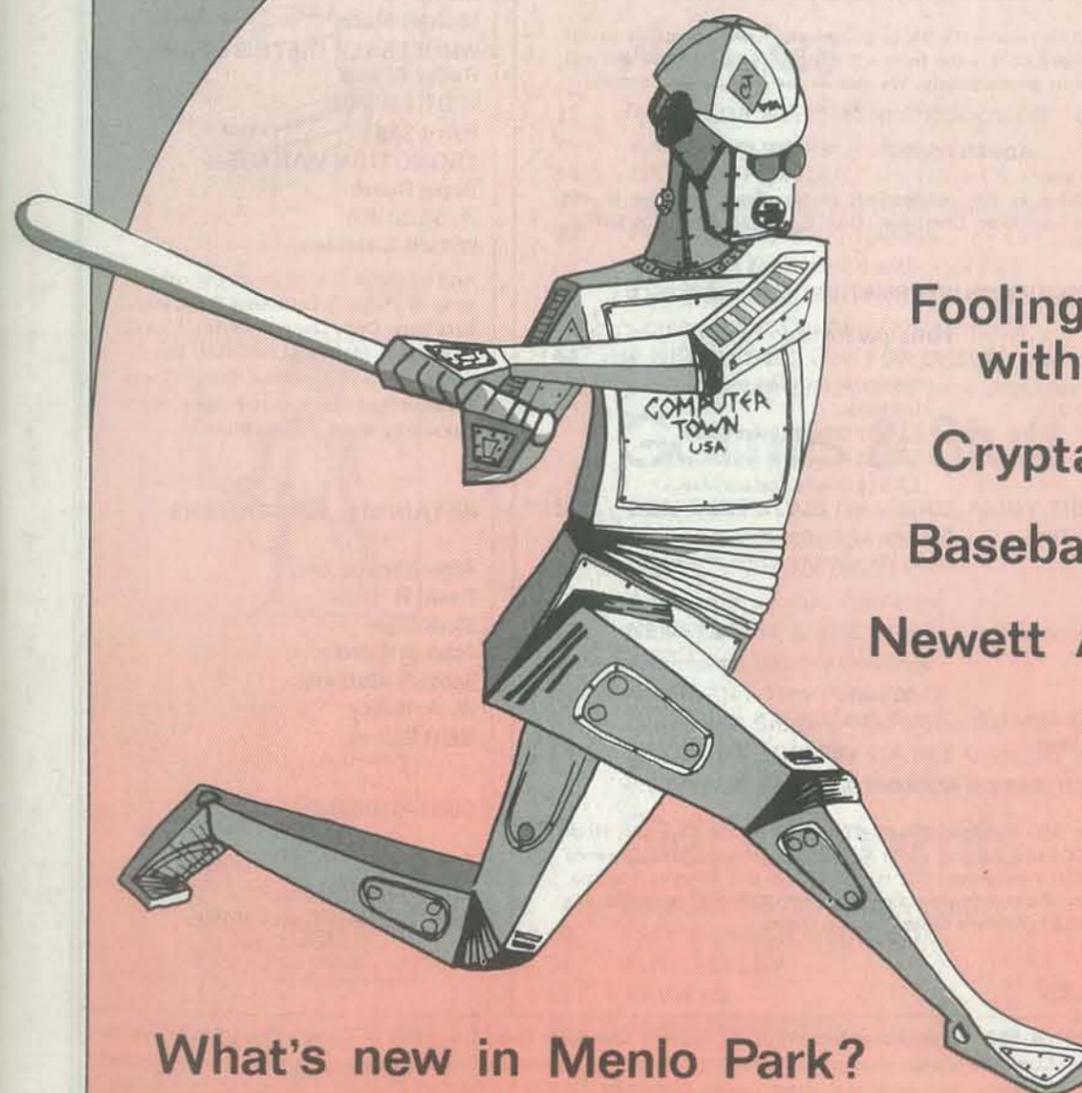
Recreational COMPUTING

VOL 8 NO 1

ISSUE 40

JULY-AUGUST 1979

SUMMER FUN



Fooling around
with your PET

Cryptarithms

Baseball

Newett Awl's Goat

What's new in Menlo Park?

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 8½x 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

ADVERTISING space is available in this publication. Please direct inquiries to the Advertising Manager, People's Computer Company, Box E, Menlo Park, CA 94025. (415) 323-3111.

SUBSCRIPTION INFORMATION

U. S. RATES

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

FOREIGN RATES

- Payments must be in \$US drawn on a US bank.
- \$17 Canada First Class
 - \$23 Rest of World Airmail
 - \$14 World Surface Mail

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

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

SAMPLE ISSUE: \$2.00

BACK ISSUES

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

FOREIGN DISTRIBUTORS OF RECREATIONAL COMPUTING

Vincent Coen, LP Enterprises, 313 Kingston Road, Ilford IG1 1PJ, Essex, UK; Rudi Hoess, Electronic Concepts PTY Ltd., Ground Floor Cambridge House, 52-58 Clarence Street, Sydney NSW 2000; ASCII Publishing, 305 HI TORIO, 5-6-7 Minami Aoyama, Minato-Ku, Tokyo 107, JAPAN; Eastern Canada: Liz Janik, RS-232, 186 Queen St. W., Suite 232, Toronto, ONT. M5V1Z1; Western Canada: Brian Wiebe.

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. Copyright © 1979 by People's Computer Company, Menlo Park, California.

STAFF

EDITORS

Bob Albrecht
 Louise Burton
 Ramon Zamora
 ART/PRODUCTION MANAGER

Sara Werry
 PRODUCTION ASSISTANT

Carole Cullenbine

ARTISTS

Aleeca Harrison

Ann Miya

Judith Wasserman

TYPESETTERS

Phyllis Adams

Gavin Cullen

Mag Glick

PROOFREADER

Nancy Heubach

CIRCULATION MANAGER

Michael Madaj

WHOLESALE DISTRIBUTION

Robin Allison

SPOT EDITOR

Harry Saal

PROMOTION MANAGER

Betsy Roeth

PUBLISHER

Willard J. Holden

And a special thanks to all the other folks at People's Computer Company: Ezra Berg, Delia Daniels, LeRoy Finkel, Loic Jassy, Mary McLean, Ann Merchberger, Paula Pustmueller, Curtis Roads, Suzanne Rodriguez, John Strawn, Nette Cekanasky Wang, Denise Winn.

RETAINING SUBSCRIBERS

Algorithmics, Inc.

David R. Dick

Mark Elgin

John B. Fried

Scott B. Guthery

W. A. Kelley

Brett Wilson

SUSTAINING SUBSCRIBERS

Bill Godbout Electronics

Byte Publications

Paul, Lori and Tom Calhoun

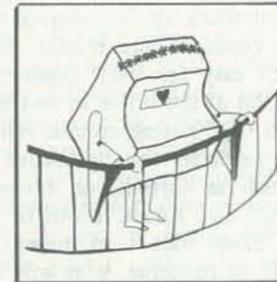
Recreational COMPUTING

Volume 8 Number 1
 July - August 1979

formerly
people's computers



pg. 10



pg. 48



pg. 24



pg. 32

COVER by Aleeca Harrison

Special Features

- 8 **CRYPTARITHMS** by John Davenport Crehore
A master puzzler poses a challenge
- 10 **COMPUTERTOWN, U.S.A.!** by Louise Burton
PCC's newest adventure - libraries, bookstores, pizza
- 32 **FORTAN MAN** by Lee Schneider & Todd Voros
Billy BASIC helps RESTORE our hero's memory
- 35 **ZORK: A COMPUTERIZED FANTASY SIMULATION GAME** by P. David Lebling, et. al.
It *is* suicide to attack a troll with a glass bottle
- 48 **WHAT LIGHT ON YONDER PANEL FLASHES?** by Ralph Roberts
Two of the fairest LEDs in all the heavens!

Articles

- 12 **THE DEDICATED WORD PROCESSOR** by Jon R. Lindsay
A Christmas message in July
- 14 **DP GAME PLAYING** by Robert S. Glass
Kareem Abdul - Jabbar in the boardroom
- 24 **PET FUN WITHOUT GAMES** by Len Lindsay
Or, how to fool around with your PET
- 28 **THE FORTE MUSIC PROGRAMMING LANGUAGE** by Jim Day
Coming soon: The Apple Philharmonic
- 44 **A NEW ALGORITHM FOR CHESS** by David Chelberg & David Watters
(Part 3) Opening and middle game strategies

Games & Stuff

- 16 **DON'T SET YOUR GRAPHICS; PRINT THEM** by Clyde Farrell
'String' together faster TRS - 80 pictures
- 18 **BBALL** by Evan Marcus
A complete baseball simulation
- 27 **NEWETT AWL & THE GOAT** by Gordon French
A microbus mystery challenge
- 51 **MATH** by Tony Pola
Computer + Student = More than Homework
- 54 **SPOT: THE SOCIETY OF PET OWNERS & TRAINERS** by Harry Saal
Products, publications, programs & praise

Departments

- 4 **EDITORS' NOTES AND LETTERS**
- 30 **PROGRAMMER'S TOOLBOX**
- 34 **FUTUREPLAY**
- 56 **REVIEWS**
- 60 **ANNOUNCEMENTS**



Editors' Notes

Lead time frustrates most magazine editors. At *RC*, we write and edit articles two to three months before our readers see them. That means depth, not timeliness, has to be our virtue. But in the next few issues (starting with September-October), we're going to print material that's fresh, that's hot, that you won't see anywhere else.

We'll be introducing you to the Atari and Texas Instruments personal computers. There will be overviews and tutorials and slick application tricks—all from the authors of the books enclosed with each Atari or TI machine. These writers are known to many of you from their previous work—Bob Albrecht, Ramon Zamora, Don Inman, Jerry Brown, and LeRoy Finkel. So tune in next issue for an exclusive look at the newest computers on the market.

Also, stay tuned to your local TV guide late this summer for the announcement of an upcoming PBS special on computers in the classroom. Called "Don't Bother Me, I'm Learning," this 60-minute documentary presents the case for computer education through the eyes of its most ardent advocates: computer-hooked kids. The program was put together by Dave Shepardson, an independent television producer in San Francisco, with several PCCers serving as consultants. Having seen the first cuts of the show, our unbiased opinion is: it's great. Don't miss it.

Our new contributors this issue come from all over—New Jersey, Massachusetts, Arizona, even Mountain View, California. One new contributor, Jack Crehore, just turned 88; another, special events critic Joanna Fried, is pushing seven years of age. That's the kind of diversity we love. Now, if only more women—besides Joanna and this editor—would write for *RC*, we'd be happy. (More on the need for a computing sisterhood in coming issues.)

Now it is time, once again, to speak of dragons. Actually, to ask you what you think, imagine, dream dragons to be. The first International Computer-Drawn Dragon contest has begun! The winner's dragon will appear on the cover of the November-December 1979 issue of *RC*. To enter this contest, you must submit: 1) the print-out of a dragon image or a photo of your screen; 2) the listing that generated said dragon; and 3) specs on the system used. The dragon can represent any mythical tradition, including the mythology inside your own head. Preference in judging will be given to friendly dragons, though ferocious ones with a touch of class will be eligible, too.

In addition to the glory of appearing on the *RC* cover, the grand prize winner will get a three-year subscription to the magazine. Second-prize is a two-year sub or 12 back issues of the winner's choosing; third prize, a one-year sub or a copy of *What to Do After You Hit Return*. DEADLINE FOR ENTERING THIS CONTEST is September 10. Send all entries to the editors, *RC*.

— Louise Burton, Bob Albrecht, Ramon Zamora.

P.S. You'll find lots of challenges in this issue. Keep reading.

Letters

DEFENDING F-MAN'S HONOR

Now don't get me wrong. I may not be the most learned man on the face of the earth, but I know when I'm being insulted! "A Different FORTRAN Man Scenario" in the Jan/Feb 79 issue was totally disgusting. To allow such an outrage to even be printed is beyond belief. And I quote: "BASIC??? He's one of the worst villains of all!!" Indeed! I thought this was the people's computer magazine. You're supposed to be catering to the software needs of the masses! Oh, of course it's easy for those bigshots who use the IBM at the office or just happen to have a CDC sitting around collecting dust, to criticize the "primitive" languages such as fortran, oh excuse me, FORTRAN and BASIC (both of which are very close friends of mine). If we allow this to continue, who will be the next to come under the gun?? LISP? PASCAL? Or even APL? That's the way it'll turn out!! We've got to draw the line! If this be the first shot, may it be heard around the world!!

*Computer Hobbyists of the
World Unite **** Fight for
Your Language!*

Your magazine has disgraced the good name of both FORTRAN and BASIC by printing this malicious slander, and I believe a formal apology to them is in order. As for the unfortunate soul that wrote the article, I can give but one word of advice: Change your ways, you decrepit being, or you may find a horse's head in your memory banks!!

CRAZYMAN Richard Brooks
65 Spring Garden Ave.
Norwich, CT 06360

P.S. Enclosed is payment for a subscription to your great magazine.
P.P.S. You have not heard the last of CRAZYMAN.

Ed. note: Nette, our keeper of accounts, verifies that Richard is crazy. He sent \$15 for his \$10 RC subscription.

GOD AND THE PET

I was surprised to see Rev. Strasma's letter in *Recreational Computing* (March-April). I am the one who wrote the "bitter terms" letter, and despite my better judgment, I just had to send a reply. I think, since he opened this bag of worms, I'll take each remark as it came.

(1) Rev. Strasma, I really don't see the point in the remark about "a non-Christian sect." Do you know what my "sect" believes in? Are you sure it isn't Christian? I'll admit I'm not a Methodist, but how did you know that? Maybe I'm a Buddhist, or a Moslem. Do you know that Moslems consider Christianity to be an infidel "sect"? I hope your neck gets over its stiffness soon.

(2) What's being a pastor, or a bricklayer, or anything else got to do with the way a company treats a consumer, and whether I can say anything about it? I say my attacks were right on! I, and many others that I know personally, have been treated in a shabby manner. Quite frankly, having my letters printed was the only way I could get Commodore to talk to me. My letters sent directly to Commodore were totally ignored.

(3) Are you sure that "luck" was the only reason your PET needed repairs twice within 90 days? Could it just possibly have been poor construction/inspection/design? Unless you dropped the PET, why weren't you reimbursed for the shipping costs you had to pay?

(4) I talked with people at Commodore too! Half didn't know anything, and the other half didn't care. I guess they figure if you've put up \$800 to buy a computer sight unseen, you can't be too bright. Maybe they're right.

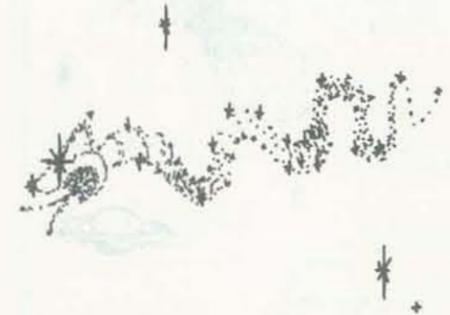
(5) I know a heck of a lot more than four PET owners, and generally we're pleased with our PETs. We are uniformly alike in our dislike of Commodore. At least the "old" Commodore of a year ago. I understand there have been some changes made around there. I wonder why?

(6) The Pet is an incredible piece of work, despite Commodore's strange marketing/quality control/secrecy in operations. I wouldn't trade it for any other micro on the market. (Well, I'd like to look a

little closer at the Sorcerer. It looks pretty nice.) Once I learned how to make my own repairs and learned how to operate it and waited until someone put out some software, it was great!

(7) I don't expect IBM service, which ain't great either. I want the same service I get when I buy an \$800 washing machine. A guarantee that works, a machine that works, and an instruction book! I *still* haven't gotten on their mailing list! They *do* have a mailing list, don't they? I was promised a mailing list. Is anyone out there? Hellooooo Commodore!

Rev. David M. Conley
Universal Life Church of the Pacific
10571 Kerrigan Court
Santee, CA 92071



HOLY WAR CONTINUES . . .

In the March-April 1979 issue of *RC* I read a poor attempt to salvage what is left of PET's pride. Unlike the Rev. Strasma of the Grace United Methodist Church, I am not biased when it comes to microcomputers: furthermore, whoever this non-Christian minister is, he is most correct in shaming the PET computer. If what you say is true about the atrocious amount of time it takes to have a PET repaired, you would be silly to buy one!

Also, are you kidding about how many times you had to have that thing repaired in 90 days? You say you're lucky? If you're lucky, then I must have had a silver spoon crammed down my throat, along with a rabbit's foot or two, a horseshoe, and a number of other assorted lucky charms.

I happen to own a TRS-80, and, since I bought it, it has not broken down. That was six months ago, six months, and not one repair necessary!! Not only that, but if it does breakdown, I have a

two-day service at *any* Radio Shack in the world! Now if those facts don't make some people convert (take that any way you want), I don't know what will.

If you had looked into microcomputers before you bought one, you might have found that the TRS-80 has disk systems already, along with printers and a number of other attachments, such as an electronic voice synthesizer, a screen printer, line printers, and many other attachments. I hope I have gotten my point across. PET owners of the world, you should have looked before you leaped . . .

Mark T. Tsetsi
38 Bourne Ave.
Tiverton, RI 02878

CHECKMATE IN FIVE!

To beat Microchess 1.5 by Peter Jennings (and also sold by Radio Shack) — in 5 moves:

E2-E4
D1-F3
F1-C4
C4-D5
F3-F7

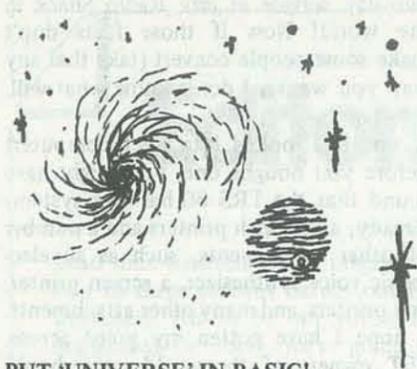
Ted Fisher
123 Marlowe
Danville, IL 61832

TEACHING TIPS, ANYONE?

I teach gifted kindergarten to fourth grade children, and I am also earning my M.A. degree at the University of Connecticut. Our school system has purchased a Radio Shack TRS-80 computer which I have been teaching the children in my math enrichment groups to program. My own interest in this area has grown, and I am therefore designing a computer unit to be used with K-8 children. Any information or materials you might have to share with me would be greatly appreciated.

Ann Doorly
SAGE Program
Northwest El. School
Hunting Lodge Road
Storrs, CT 06260

First, subscribe to RC. Then, wait for letters from some of our other teacher-readers. (Go on, drop Ann a line!)



PUT 'UNIVERSE' IN BASIC!

I believe that Les La Zar ("Universe—An Immodest Proposal," *RC*, March-April 1979) was much too hasty in writing off BASIC as the medium for his game. While one cannot argue with its limitations, it is nevertheless true that almost all micro users have BASIC running on their systems, and most would be unwilling or unable to lay out \$300-500 for FORT-RAN or PASCAL, not to mention the disk system which is virtually necessary for both of those languages (though the disk is very desirable to keep reasonable the time necessary to load "UNIVERSE" overlays). Also, many of us are having a hard enough time becoming proficient in writing BASIC programs without having to learn a new language.

The more people who contribute to "UNIVERSE," the better the chance it will actually be implemented. Therefore, let's all get busy writing fantasy games, in BASIC, and share the results. Ideas such as "UNIVERSE" are too good to let die because people have been frightened away from them. "ADVENTURE" and "DUNGEONS AND DRAGONS" were handed down by the God of the Big Computers, but we little folk can create their successors.

Incidentally, the power of BASIC can be greatly extended through the use of machine language subroutine calls, which are supported by most current BASICs. Unfortunately, this restricts the program to a particular microprocessor, but if carefully documented, these short segments can be recorded with relative ease.

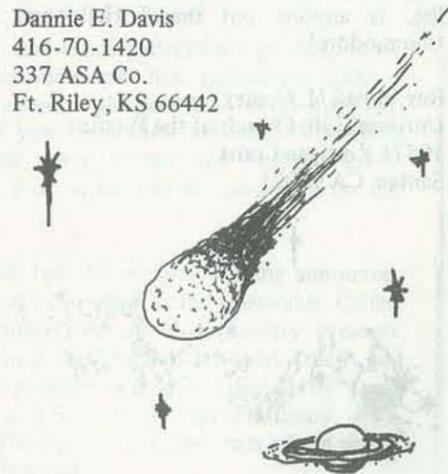
So, let's stop the sterile debate over language, and all get started writing some good games, in BASIC.

Tom Burke
R.D. 2
Fairfax, VT 05454

RELEVANCE OF 'UNIVERSE'?

In reference to "Universe—an Immodest Proposal," by Les La Zar, the difference between playing "Universe" and participating in reality is the end result. We have one hundred times the computational power of the 1950 "computers"—that simulated the thermonuclear detonation—in our personal computers. Why settle for a game when it is within our reach to affect our reality in a positive and personal way?

Dannie E. Davis
416-70-1420
337 ASA Co.
Ft. Riley, KS 66442



YEA, 'UNIVERSE'! NAY, TOLKIEN!

This letter concerns your March/April issue. I have subscribed to *RC* and *People's* for nearly two years and have not felt the urge to get in touch until now. This issue has both the best and the worst of what I read it for. The best is the article on the game *Universe* by Les Lazar. I think the idea is well thought out, the problems involved are carefully articulated and, best of all, *the game is interesting*. I enjoy reading about gaming in a context other than that of a 12-year-old, a stage which many of your writers have obviously never outgrown.

Escapist entertainment, which most of fantasy writing is, has nothing to say to me. *Runequest*, for instance, is complicated but shallow. *Universe* is the first simulation game which I have read about that seems to have the possibility of taking into account the complexity of human motivations. Whether one is writing fantasy or anything else, to leave out the human element leaves a work which is simply inhuman.



This brings me to the point of commenting on your film reviews of *Lord of the Rings*. I saw the film recently and hated it. It was a cartoon as opposed to an animated film, and I told people this. The Tolkien *cognoscenti* replied that one has to have read the books to understand the film (which I thought was a hopeless mishmash and told these people so). One doesn't have to read anything to go to a movie and either enjoy it or hate it. The work of art, whatever it is, stands on its own merits.

What is one to make, for instance, of a race of beings (Orcs), a bunch of half-backs in Halloween masks, with so little reason for existence—or so evil—that it should be totally destroyed? The other race, the Hobbits and wizards etc., are of such a superior nature that no question is posed about the righteousness of their acts. This, I suppose, is the master race. Then what are those gray things with the horns? Jews? Viet Cong?

The Orc race possesses a language in common with the Hobbits and has at least enough of a culture to bind it together, as do the Hobbits and wizards. But we are given to believe that the master race is entitled to the power (ring).

Is this getting through at all? I hope the truth of the symbolism is coming home to your readers. *Lord of the Rings* offended me on every level which I can possibly be offended. The worst offense is the fact that the film bored me. The animation was pretentious and sloppy (one can find better on any of the Levis commercials which use the same techniques). The characterization of supposedly adult individuals was shallow. It does not matter that these creatures are supposed to be human. It is simply that they act blindly and without motivation and do not question what they do.

Lord of the Rings, then, is the perfect place to take your date before heading down to the local disco for some more vacuous entertainment.

I guess the thing that bothers me most about the movie and the readers of the books is the attitude of reverential respect both for the source material and for the film made from it. To put it another way, if one does say something against it, then one obviously does not know what one is talking about. Such smugness is found in great, nay, *humongous* abundance among the readers of Mr. Tolkien's books.

The whole genre of so-called sword and sorcery literature—big, muscular barbarians and their big, muscular, barbaric women—is vile. An altogether appropriate critique of this stuff can be found in the afterword to a novel called *The Iron Dream* by Norman Spinrad. Find a copy of the book, and read it for yourself. I guarantee you will be surprised.

There are, on the other hand, numerous examples of good writing about the fantastical. Michael Moorcock's *Dancers at the End of Time* trilogy is one of the more outstanding examples. Another is *The Devil and W. Kaspar* by Benjamin Appel. And, of course, there are Phil Farmer's Riverworld books. I rest my case.

An ideal vehicle for the implementation of *Universe* is that contained in Frederick Pohl's novel, *Gateway*, in which an alien race's ships are found and can be traveled in, but the crew never knows what the destination of these ships will be. Some journeys are hazardous, while others can make the voyager rich.

Another extremely rich source of continuing adventure material is the series of stories and novels of the Eight Worlds by John Varley, especially his collection titled *The Persistence of Vision*. Both Pohl and Varley write about people, real people, confronting other real people, with the growth of both parties the result.

Einstein told us that God does not play dice with the universe. And to play at dice with one's life is equally absurd. So let's have a lot less of the likes of *Runequest* and *D&D* etc. in *Recreational Computing*. Let's explore the possibilities of *Universe*.

Lon Ponschock
203 S. Douglas St.
Appleton, WI 54911

WHAT'S IT SAYING?

Help !! We have a DEC PDP 11v03 Computer system, on which we are running RT-11, version 2, and MUBAS, version 1 (single job monitor).

We would like to be able to run four floppy diskette drives, instead of just the two that are part of the original system.

We have the equipment, have made patches to the monitor, and, in fact, are able to handle all four floppies in the usual way with PIP. However, error

messages ?DEV and ?DNE result from all attempts to access the extra 2 diskette drives in MUBASIC.

DEC says that it appears that what we are doing should work.

Any and all non-obscene suggestions would be most welcome.

Signed : Desperate
Leslie R. Tanner
Mathematics Department
Jamestown College
Jamestown, ND 58401



Translatus Uggliqr is upset

Assume that x is in $\%ZQ/$, $y = \%ZQ, 1/$ $z = \%ZQ, 2/$
 Now define P thus: $\%dt, P, \%li, \%ZQ/3, I, D/// =$
 and voila! $\%Z\%P/1, 1/\%Z\%P/2/ =$

From the Land of Sam

P.S. Typing Effort: SAM76 - 42 + ϕ
 + * line numbering or format DRAGONSIC - 53 + 8*
 See *R.C.* - Mar/Apr 79 - Page 51 BASIC - 81 + 28*
 PASCAL - 83 + ϕ + 13

From the land of SAM - Recursive Factorial

first define the procedure "fac"

$\%dt, fac, \%ig, 1, q1, 1, \%mu, q1, \%fac, \%su, q1, 1 /// =$

then convert the "q1"s into holes for variables

$\%pt, fac, q1/ =$

now get factorial of 500 thus:

$\%fac, 500/ =$

See page 27 - *RC* - march/Apr 79 and compare typing effort (69/SAM76)

Cryptarithms

BY JOHN DAVENPORT CREHORE (NINE HEX)

Jack just celebrated his 88th birthday. As part of the celebration, we are pleased to begin a series of puzzles that he devises called "cryptarithms"—a sort of crossword puzzle in arithmetic.

If you are like me, you have probably shied away from solving these kinds of problems. But Jack is both engaging and convincing; he now has me hooked. With this series, he will lead you into the brainteasing world of cryptic arithmetic. As an added challenge, he suggests that we keep a running tally of those of you who respond. We will publish your initials (or puzzler name—Jack's is Nine Hex) and the totals for the number of problems you solve. All problems will be numbered sequentially for use in all communications.

In addition, Jack poses problems in several categories: Novice, Adept, Genius, and Computer. We want to see and publish any programs you may develop for solving this class of logical

problem. In a later article Jack will lead you through a variety of solution schemes for arriving at answers to the puzzles.

"I am one of the oldest timers—an Aficionado, a Buff, a Master—in a select cult of puzzle solvers who dabble in math from kindergarten to the doctoral level," Jack notes in his letters to us. "Thirty or forty years ago, for a few months, I ran a half-page in one of the Mechanics magazines. About then, the international societies for the elite cryptographers—the American Cryptogram Association and The National Puzzlers' League—were in their heyday. I enjoyed the tutelage of many of the skilled cryptographers of our military and state departments," Jack concludes.

Over the years, Jack has continued to perfect his interest in puzzles. He has taught puzzle-solving to both children and adults. You now get a chance to enjoy the benefits of his expertise.

—RZ

SAMPLES & EXAMPLES

Substitute the correct numbers (digits) for the letters and you will solve some simple examples. You can begin by determining the range of values for each letter. Is it more or less than the number five? Is it odd or even?

Puzzle 1

$$\begin{array}{r} B \\ B \\ B \\ \hline CB \end{array}$$

Puzzle 2

$$\begin{array}{r} R \\ R \\ T \\ \hline TR \end{array}$$

Puzzle 3

$$\begin{array}{r} AC \\ \times AC \\ \hline FH \\ JF \\ \hline YBCH \end{array}$$

Puzzle 4

$$\begin{array}{l} A, ABB, FCB \\ +H, KCF, FKB \\ \hline HM, MAM, MFA \end{array}$$

CHALLENGES

Puzzle 5 (NOVICE)

$$\begin{array}{r} KM \\ H \\ M \\ \hline MDD \end{array}$$

Hint: Look at the samples & examples

Puzzle 6 (ADEPT)

$$\begin{array}{r} NADUES \\ \times NADUES \\ \hline HHANDBS \\ NADUES \\ EBDHUAD \\ NDTTEHA \\ EETEUTA \\ \hline ETTYBHTHSHUS \end{array}$$

Hints: (a) S X S = S! (b) Letters for one and zero are obvious. (c) The units columns of the five products show three different digits. (d) The fifth product contains a perfect square— N^2 . (e) The third product has a zero in the units position, so a five is involved.

Puzzle 7 (GENIUSES)

$$\begin{array}{r} YER LUS NAS \\ YYB LUM UAL \\ YUR LUR UAR \\ \hline LNR ASR UMR \end{array}$$

Hints: No hints to geniuses! They compete on even terms with computers!

Puzzle 8 (COMPUTERS)

$$\begin{array}{r} HL ECD DTB \\ \times LM UWT \\ \hline DL BMT BCW \\ CEL WWL CB \\ HELWW HEB D \\ HBEULL DLW \\ ULDUBH WE \\ \hline MMWCCD HEU WHW \end{array}$$

Hint: No hints here, either! A genius can solve this, in time.

So try your skills and get your answers and programs to us as soon as possible. Again, in future issues we will keep a running tally, under your initials or name, based on your correct answers. So bend your brain and get those letters rolling. As Jack (also a CB fan) says in his notes to us, "Seventy-threes, good buddy. Eights." □

SOLUTIONS TO SAMPLES AND EXAMPLES

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.

$$\begin{array}{r} 10,080,058 \\ + 1,235,524 \\ \hline 8,844,534 \\ \hline 10,080,058 \end{array}$$

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.

$$\begin{array}{r} 1024 \\ \times 32 \\ \hline 32 \\ 64 \\ \hline 96 \\ \hline 1024 \end{array}$$

The number 9 is the only digit that works.

$$\begin{array}{r} 19 \\ \times 9 \\ \hline 9 \\ 9 \\ \hline 19 \end{array}$$

The number 5 is the only digit that works.

$$\begin{array}{r} 15 \\ \times 5 \\ \hline 5 \\ 5 \\ \hline 15 \end{array}$$

COMPUTERTOWN, U.S.A.!



BY LOUISE BURTON

A dot flashes in the center of an unidentified state map. The computer waits. The name of the state is . . . ? The nine-year-old at the keyboard wiggles, then turns questioningly to her mother. "Well," Mom says, "where did Dorothy live in *The Wizard of Oz*?"

"Kansas! Kansas!" Enlightenment. She types the word and hits ENTER. The next state's outline appears on the screen. A crowd of kibitzers gathers behind her to share in this computer game. It is Tuesday night in Menlo Park, California, and the place is the public library.

A humanities professor sits down at a computer and threads his way past the monsters in a dungeon. Strange, very strange, he keeps thinking. He had simply wandered into one of his regular literary hang-outs, and there was this group of people playing with computers. And now he was trying it too. It is Friday night in Menlo Park, California, and the place is Kepler's Bookstore.

Monday night is the Smiths' night out, and by family vote, dinner is pizza. What do they discover then they arrive at their local pizza parlor? Several picnic tables given over to computers. And where is it happening? Why, Menlo Park, of course.



Menlo Park, California, home of People's Computer Company, may soon be known as *Computertown, U.S.A.* Grass-roots computing has caught on here. Every month, it seems, computers spring up somewhere new.

In February, PCC put on a "computer carnival" at Kepler's Bookstore, a large and popular establishment on El Camino Real. The event, advertised in the local newspaper and by banners in the store windows, drew a mixed crowd of computer believers and skeptics, kids and parents, book browsers and folks just off the street. Kepler's now has a computer carnival the third Friday of each month, run by the Dragons of Menlo Park (otherwise known as Bob Albrecht, Ramon Zamora, and friends).

In March, the dragons brought their computers to the Menlo Park Public Library, launching a weekly computer night that is now a Tuesday tradition. After the first night, the sessions were mobbed. Kids greatly outnumber adults at the library gatherings, and those with several weeks' experience are teaching their buddies.



Seeing this enthusiastic response, Commodore Business Machines of Palo Alto loaned a PET computer to the library full-time. It is now in residence in the children's section. To use the PET, youngsters must attend a one-hour class, also led by the PCC team. The kids practice loading cassettes, typing commands, and playing a game. Each graduate receives a "My Computer Likes Me" button and is eligible to use the PET whenever the library is open.

In May, Peninsula School in Menlo Park held its annual spring Learning Fair (see "review" section). As at past fairs, there were computers on hand. But this year there were eight machines, with eight different programs running simultaneously, thanks to the Cluster/One disk system provided by RC columnist Harry Saal.



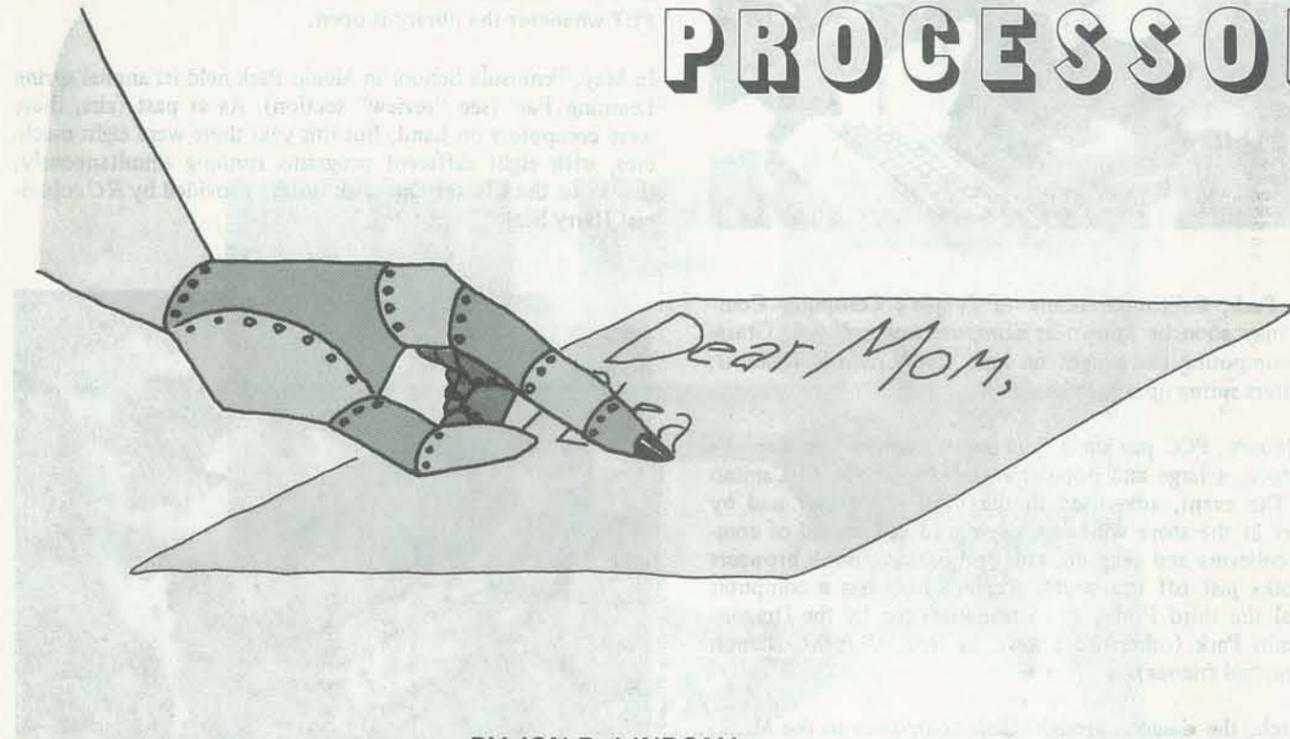
In June, monthly computer nights (like those at Kepler's) started at Round Table Pizza in Menlo Park, a casual family restaurant.

These recent events, claims Bob Albrecht, are just a beginning. "We've talked for a long time about PCC being a community resource," he says. "And now we're making it happen — taking computers to the places where people have fun. It's not enough to tell people that computers aren't intimidating. You've got to *show* them that it's true — in a relaxed environment. What's starting to happen here in Menlo Park is that a town of 27,000 is turning into one big learning center. And computers are only part of it."

Perhaps the best plug for the recent programs came from a library-class alumnus, Lauren Miller, age 11. "It's a good thing," said Lauren, "to use computers so young. 'Cause when you grow up, there's going to be millions of 'em. I think they'll even be more common than TV!"

The photos above were taken at the Menlo Park Public Library.

THE DEDICATED WORD PROCESSOR



BY JON R. LINDSAY

Microprocessors and personal computers must be taking over the world. They have reached Fresno, California, so can the rest of the world be far behind? Jon proves they are in Fresno, and working there, by sending us this article on how he uses his computer as a word-processing device.

Of course, he could be fooling us; playing on our sympathy for hobbyists who struggle to compute in small towns across the country. (I know I'll probably get a flood of letters—two or three, at least—from Fresnoians defending the size and cosmopolitan qualities of their city.) He probably lives in Los Angeles, in a penthouse condominium, surrounded by medium-sized computers. Why not write to him and see if he (if anyone) lives in Fresno. His address is 3812 N. First St., Fresno, CA 93726. —RZ

There are word-processors and then there are word-processors. The good ones are designed to make creating word text as easy as possible and to speed the production of printed material along. I use a simple form of word-processing for creating text, e.g. letters, forms, reports, and lists.

But what if you already have the text in mind? Then the emphasis shifts from creating to producing. Suppose you wish to send the same letter to many people. The problem becomes the typing of the letter and the envelope.

The simple program I have written is dedicated to producing just one letter form at a time. But that letter form can go to hundreds of individuals—with a personal touch. It is here that this program outperforms a mimeograph machine. All of the information is stored within the program.

The first aspect of the program—the text of the letter—is inserted starting at line 180. As long as only the content of the letter is varied, the program can be used for any number of other letter forms. What we lose in ease of editing is made up for by the simplicity of the program.

The example used in the program was a Christmas note to business contacts. Their names and addresses are in the DATA section and include four string variables. The first, G\$, is the first name or nickname. When the relationship exists for such informality, it's a natural and friendly letter that greets the reader.

The remaining variables, N\$, A\$, C\$, are the components of the address. I didn't need to include any corporation or company heading, but this could be easily

```

10 REM          PROGRAM NAME < LETAD >
20 REM          PROGRAM FUNCTION < DEDICATED WORD-PROCESSOR >
30 REM          BY Jon R. Lindsay
40 REM          EQUIPMENT:
41 REM          Z-80 CPU
42 REM          SOROC-120
43 REM          DIABLO HY-TERM PRINTER
44 REM          ICOM DUAL DISK DRIVES
45 REM          MICROSOFT (CP/M Based) DISK EXTENDED BASIC
50 WIDTH 120
60 PRINT CHR$(27);CHR$(43);REM CLEAR SCREEN
70 PRINT "Select operation - 1. LETTER"
80 PRINT "                        2. ADDRESS"
90 INPUT E
100 IF E<1 OR E>2 THEN 60
110 ON E GOTO 120,460
120 READ GS,NS,AS,CS;REM GET DATA FROM DATA LIST
130 IF GS="END" THEN 60
140 PRINT "Insert letterhead and center it. Then HIT RETURN";INPUT E
150 REM
160 REM          CONTENT OF LETTER
170 REM
180 LPRINT:LPRINT:LPRINT:LPRINT:LPRINT:LPRINT
190 LPRINT TAB(25);"December 15, 1978"
200 LPRINT:LPRINT:LPRINT:LPRINT
210 LPRINT NS
220 LPRINT AS
230 LPRINT CS
240 LPRINT LPRINT
250 LPRINT "Dear ";GS;": "
260 LPRINT LPRINT
270 LPRINT "The holiday season is a time to renew old friendships and to say"
280 LPRINT CHR$(34);"thank you";CHR$(34);
290 LPRINT " for the support you and your staff have given us in"
300 LPRINT " the past."
310 LPRINT LPRINT
320 LPRINT GS; ", we sincerely appreciate taking care of your accounts. We"
330 LPRINT "are very interested in continuing to give your patrons the same"
340 LPRINT "thoughtful care they receive in your office. This is very import-"
350 LPRINT "ant to us."
360 LPRINT LPRINT
370 LPRINT "Again, please accept our wishes to you and your staff for a Happy"
380 LPRINT "Holiday Season and a Great New Year."
390 LPRINT LPRINT
400 LPRINT TAB(40);"Sincerely,"
410 REM
420 REM          END OF LETTER
430 REM
440 LPRINT CHR$(12);REM LINE FEED TO SCROLL PAPER OUT OF MACHINE
450 GOTO 120
460 PRINT CHR$(27);CHR$(43)
470 PRINT "Do you want envelopes addressed? < Y = YES >";INPUT Y$
480 IF Y$="Y" THEN 60
490 RESTORE: REM ALLOWS REREAD OF DATA LIST
500 GOTO 570
510 READ GS,NS,AS,CS
520 IF CS="END" THEN 60
530 LPRINT NS
540 LPRINT AS
550 LPRINT CS
560 LPRINT CHR$(12)
570 PRINT "Insert new envelop and center. Then HIT RETURN";WAIT 0,1,1
580 GOTO 510
590 DATA "Hugh","Hugh A. Schafter, D.D.S.", "447 N. First Street, Suite 170"
600 DATA "Fresno, CA 93726"
610 DATA "Buzz", "Randolph Meyer, M.D.", "3460 N. Fresno St., Suite 45"
620 DATA "Fresno, CA 93713"
630 DATA "B.K.", "Bernard K. Karlan, Ph.D.", "1250 W. Apple Ave."
640 DATA "Fresno, CA 93711"
650 DATA "ETC.", "ETC.", "ETC.", "ETC."
660 DATA "END",,,
    
```

done by inserting another string variable between N\$ and A\$ and then including the company name in the data list. Remember to make space for this variable for each person on the data list.

A word about the data list: since the people I want to send such a letter to are the ones I deal with most often, I rarely manipulate the list, except to add names. Once this list has been developed, you can use it again and again, only changing the letter content.

The second aspect of the program is the envelope addressing. In line 490 the data is re-stored for rereading and the process is repeated. Only the address string variables are utilized. Lines 440 and 560 are printer line feeds to scroll the letterhead or envelope out of the machine, making

way for another. This assumes you load each piece separately. It isn't necessary, but is handy.

In line 280 I wanted to enclose "thank you" in quotes, but Microsoft Disk Basic (like others) doesn't allow that within the string literal, unless it is the beginning or termination of the string literal. An easy way around this problem is to print CHR\$(34) in the appropriate places to generate the quote mark. It gives the letter a more natural appearance and, I think, is superior to the use of an apostrophe.

Study the program, adding to or deleting from it as you see fit. I think you'll find it a handy addition to your program file. □

December 15, 1978

Randolph Meyer, M.D.
3460 N. Fresno St., Suite 45
Fresno, CA 93713

Dear Buzz:

The holiday season is a time to renew old friendships and to say "thank you" for the support you and your staff have given us in the past.

Buzz, we sincerely appreciate taking care of your accounts. We are very interested in continuing to give your patrons the same thoughtful care they receive in your office. This is very important to us.

Again, please accept our wishes to you and your staff for a Happy Holiday Season and a Great New Year.

Sincerely,

Randolph Meyer, M.D.
3460 N. Fresno St., Suite 45
Fresno, CA 93713

D P G A M E P L A Y I N G

HE JUST COULDN'T QUIT

BY ROBERT S. GLASS

Rick Barry passes to the Vice President! The VP dribbles the ball and fires a bounce pass to the Comptroller! The Comptroller sinks a 16-foot jump shot! The game is tied 32-32! Huh?

It's a game. It's a simulation. It's both! Read on and enjoy this fanciful (or is it?) article.

—RZ

By now, everybody knows about computerized games. From Star Trek to Land War, the concepts and actions associated with playing against or with an electronic beastie have intrigued game-players from Poughkeepsie to Podunk.

That kind of game-playing is usually an in-the-privacy-of-your-home (or retail computer store) kind of thing. Sometimes—perhaps more often than anyone cares to admit—that kind of game playing also occurs time-shared with other, more productively occupied, computer tasks.

And occasionally—perish the thought—game players actually squeeze a few sessions in around the rough edges of an 8 to 4:30 work day. On company time, in short.

David Blast was one of those. Disguised as a production programmer for General MPG, Dave was actually an impish game player who preferred doing battle with Klingons to doing battle with Cobol.

If there was a game to be played on the Marketronics 3PI computer at General MPG's corporate computer center—even a bootlegged one—Dave knew about it. And played it.

Dave resented—sometimes deeply—the fact that his surreptitious sessions were often prematurely terminated by the arrival of a curious manager whose brows furrowed when he saw the unusual displays being generated on Dave's supposedly Cobol-productive CRT screen. With the last of the red-hot Klingons locked in your sights, it takes a person of psychological steel to snap off the power switch.

The problem, Dave felt, had to be solved. And with all the system skills of a knowledgeable programmer, he set about finding a solution.

The responsible, pragmatic soul in all of us probably demands that one obvious solution be considered—Quit Playing Games. It's the right and proper answer to an essentially ethical dilemma.



Dave, as you may have already guessed, quickly rejected that on the grounds that it was No Fun at All. So much for ethics.

Another much more attractive answer evolved slowly as Dave worked the problem back and forth in his mind. The problem, succinctly stated, was Avoid Getting Caught.

Played against this backdrop of anti-institutionalized thinking was another theme on which Dave had been working: computerized basketball. Why not try out his Avoid Getting Caught philosophy and build a new game at the same time? The idea was no sooner formulated than it was set to coding pad.

The essence of the output of Dave's basketball game was a dynamic output display containing the roster of players for each of the two competing teams. As the game progressed, the human managers at the console could select who started the game, who was substituted and when and which offensive and defensive strategies were to be employed.

The computer, upon receipt of this controlling input, would simulate a result of the offensive/defensive strategies, ultimately recorded in real-time as a score or a turnover or a foul or some other basketball-relevant interim result.

A FLOWER BLOOMS

But all of that was fairly obvious basketball output. The curious manager could crucify a guy caught with one of those displays on his screen. How, in fact, could Dave even begin to think he could make a computerized basketball game look like something else?

It is from dilemmas such as these that the flower of true creativity blossoms. The basketball game would continue to play, its output openly displayed before God and Country, with no one the wiser. Its output would simply be transliterated into a superficially sanctifiable, transparent-to-the-player disguise.

An output encryption scheme where both the message and its coded form were meaningful.

A management simulation game. The output would take the form of a management simulation game. Where the rosters would be organization charts. And each box on the chart would represent a player.

Subordinate to that box would be his potential substitutes. In the box, with the player, coded as an organization number, would be his current data—points, fouls, anything else pertinent. And across the bottom of the screen, proclaimed as a running simulation tally, would be the encoded game score.

THE SYSTEM WORKS

"What are you doing?" an alert manager would say to Dave.

"Checking out a management simulation algorithm for Industrial Relations," would be Dave's equally alert reply.

Well, the system worked. When I last talked to Dave, he had a six-person league going, playing basketball every day, according to a predefined 40-game schedule. □



**this
publication is
available in
microform**



Please send me additional information.

University Microfilms

International

300 North Zeeb Road
Dept. P.R.
Ann Arbor, MI 48106
U.S.A.

18 Bedford Row
Dept. P.R.
London, WC1R 4EJ
England

Name _____
Institution _____
Street _____
City _____
State _____ Zip _____

Don't SET your graphics;

PRINT them

Clyde, a producer of exceptional TRS-80 software, sends along the following graphics utility. The program helps you create and save combinations of graphics characters that you may wish to use in some of your programs.

I am sure he has used this routine to put together the Klingon and Enterprise ships in his Trek games. For information on his software write to Farrell Enterprises, P.O. Box 4392, Walnut Creek, CA 94596. -RZ

BY CLYDE FARRELL

Anyone who has had their TRS-80 for longer than just a few months has surely found that using SET and RESET for graphics is both slow and difficult. When the LEVEL II manual came along, we were advised to use POKE to speed things up a bit, which it does . . . but have you ever POKEd to the wrong location accidentally?? The results can be devastating! So what's a programmer to do?? I think that the solution might be found in CHR\$ graphics.

The concept of CHR\$ graphics may be new to some, so a little explanation may be necessary. All the characters in your TRS-80 have an ASCII representation. This also includes the graphics characters. For example, if you type:

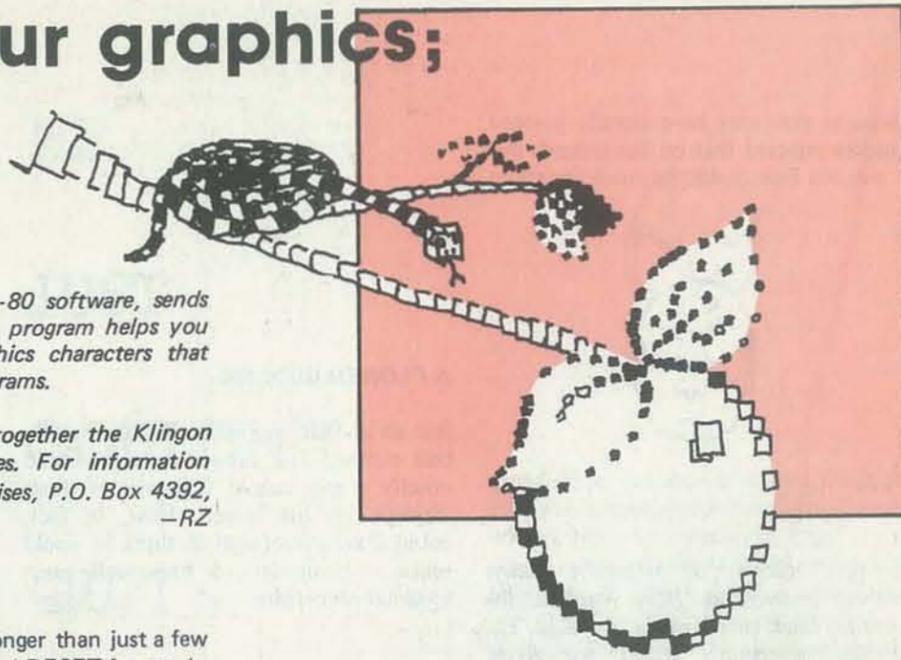
```
>PRINT CHR$(48)
```

and press ENTER then a zero will appear on your screen. Zero has the ASCII value of 48. If, on the other hand, you type:

```
>PRINT CHR$(135)
```

and press ENTER you get something quite different. A graphics character will appear on your screen. With this idea in mind, write a little program like:

```
10 A$ =
20 FOR X = 1 TO 3
30 A$ = A$ + CHR$(138)
40 NEXT X
50 PRINT A$
```



Run it, and you will see a nifty little splotch that is actually three graphics characters all strung together. If you took the time to string together enough of the proper CHR\$ values, you could create actual pictures (dice, the Enterprise, etc.) that could be "drawn" on your screen almost instantaneously by using a PRINT statement. Just think of the ease in time, effort, and memory. A convenient way to create these little pictures would be with some type of a utility program that would allow you to enter the ASCII value of a graphic character, display the character, append it to the string of characters you are building, allow you to "erase" it if you had picked the wrong one, and then record it (either on tape, disk, or at least give you the lines of BASIC necessary to recreate the character in another program without having to go through the strain all over again).

The graphics utility presented here does all of these things. A picture can be built using any character (graphics or otherwise) and dynamically changed as the building process continues. Erasing the previous character is accomplished by entering '0' as the ASCII value you are prompted for. If you are not sure which graphics character you need, entering a negative number will display a screen of all the valid graphic characters and their corresponding ASCII values. Once the final shape has been created, entering '999' will give you the opportunity to both see the lines of BASIC necessary to recreate the 'character' in another program and you can record the 'character' on a data file. The program is written to record the character on tape, but a small modification would allow disk retention.

As you build your graphics display, a '#' symbol will show as your 'cursor.' It will indicate to you what your present position on the screen is, but keep in mind that a graphics character actually extends below the print line for 'regular' characters.

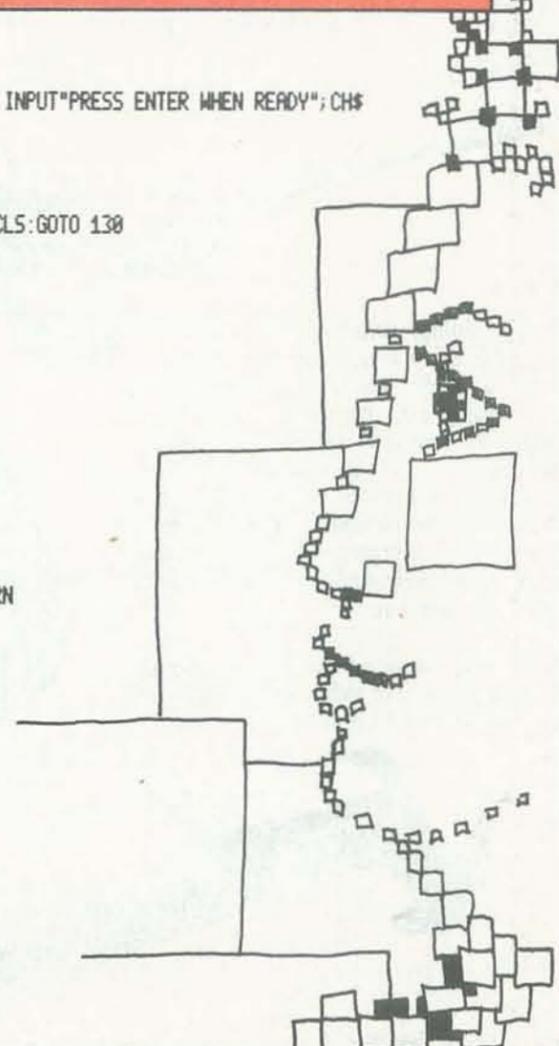
Use this program to create some graphics representations and then try PRINTing them instead of going through all the SET and RESET hassles. I'm sure you'll find this method draws them faster and safer, and it's a lot more fun to do.

Here is a list of useful ASCII values that are not in the graphics display screen but are still very useful in composing a 'drawing.'

ASCII VALUE	EFFECT
26	LINEFEED (Does not erase to end of line)
24	BACKSPACE
128	SPACE (BLANK)
27	UPWARD LINEFEED

Other ASCII codes and their uses can be found in the appendices of your LEVEL II manual.

```
5 CLEAR 700: DIM CL(255)
10 CLS: INPUT "DO YOU WANT A TAPE DATA FILE"; YN$
20 IF LEFT$(YN$, 1) = "Y" THEN PRINT "PREPARE TAPE IN RECORDER": INPUT "PRESS ENTER WHEN READY"; CH$
40 CLS
50 CH$ = "": X = 1: GOTO 130
60 PRINT@832: INPUT "WHAT ASCII CODE"; CN
70 IF CN = 0 THEN IF X > 1 THEN X = X - 1: CH$ = LEFT$(CH$, LEN(CH$) - 1): CLS: GOTO 130
80 IF CN = 999 THEN GOSUB 2000: GOTO 50
90 IF CN < 0 THEN GOSUB 1000: GOTO 130
100 CLS
110 IF CN > 191 THEN PRINT "INVALID CHARACTER": GOTO 130
120 CH$ = CH$ + CHR$(CN): CL(X) = CN: X = X + 1
130 PRINT@200: CH$ + "#"
140 GOTO 60
1000 CLS
1010 FOR Y = 129 TO 191
1020 PRINT Y; CHR$(Y);
1030 NEXT Y
1040 A$ = "": A$ = INKEY$: IF A$ = "" THEN 1040 ELSE CLS: RETURN
2000 CLS
2010 PRINT: PRINT: PRINT "FOR X = 1 TO"; X - 1
2020 PRINT "READ A$";
2030 PRINT "VARIABLE$ = VARIABLE$ + A$";
2040 PRINT "NEXT X";
2050 PRINT "DATA ";
2060 FOR Y = 1 TO X - 2
2070 PRINT CL(Y); ", ";
2080 NEXT Y
2090 PRINT CL(X - 1)
2100 IF LEFT$(YN$, 1) = "Y" THEN PRINT#-1, CH$
2110 PRINT: PRINT: PRINT "PRESS ENTER FOR ANOTHER CHARACTER";
2120 INPUT CH$
2130 CLS: RETURN
```



BBALL

A Baseball Simulation

BY EVAN MARCUS

Evan lives at 117 Manning Ave., River Edge, NJ 07661. He indicates that he and his friends at River Dell High School in Oradell, NJ, have many more programs developed. I am sure that we will hear more from Evan and his classmates in the future. - RZ

This program is a complex example of random competition. You are pitted against the computer in a theoretically even game. If you use all available strate-

The game is different from many other computerized baseball games in that it doesn't ask you for each pitch you wish to throw. Instead, you simply state that you wish to be pitched to.

In effect, there are just a couple of limitations on the game. Injuries do not occur; there can be no arbitrary rain-outs; and once a player gets past first base, he either scores or is stranded (unless he is caught stealing). The computer can only steal 2nd base, unless the hit-and-run is on, while you can steal any base. The chance of stealing is related to the square of the base you attempt to steal.

The only time that the computer plots a strategy is before the first pitch, while you can do it at any time.

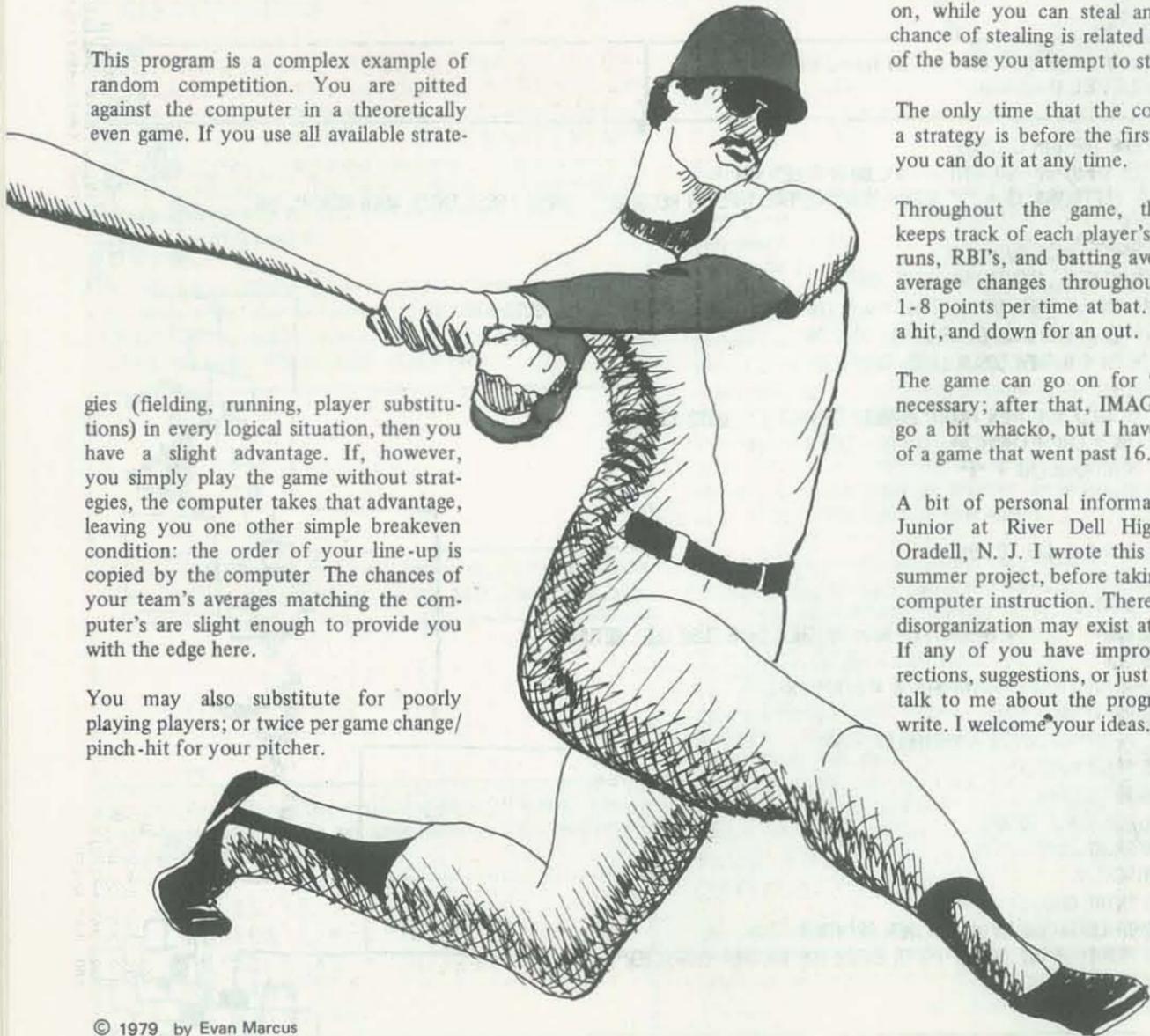
Throughout the game, the computer keeps track of each player's at bats, hits, runs, RBI's, and batting average. Batting average changes throughout the game, 1-8 points per time at bat. It goes up for a hit, and down for an out.

The game can go on for 99 innings if necessary; after that, IMAGE statements go a bit whacko, but I have never heard of a game that went past 16.

A bit of personal information. I am a Junior at River Dell High School in Oradell, N. J. I wrote this program as a summer project, before taking any formal computer instruction. Therefore, a bit of disorganization may exist at some points. If any of you have improvements, corrections, suggestions, or just would like to talk to me about the program, PLEASE write. I welcome your ideas. □

gies (fielding, running, player substitutions) in every logical situation, then you have a slight advantage. If, however, you simply play the game without strategies, the computer takes that advantage, leaving you one other simple breakeven condition: the order of your line-up is copied by the computer. The chances of your team's averages matching the computer's are slight enough to provide you with the edge here.

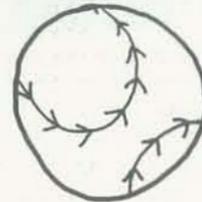
You may also substitute for poorly playing players; or twice per game change/pinch-hit for your pitcher.



© 1979 by Evan Marcus

BBALL

Listing



```

10 REM
20 REM
30 REM
40 REM
50 REM
60 DIM A(18,9),ASC(21),B(5),L(20)
70 DEF FNAC(X)=INT(RND(1)*X)+1
80 DEF FNB(Y)=Y+9*(TO=2)
90 DEF FNC(Z)=ALFN(B(LLTOJ),Z)
100 PRINT "INSTRUCTIONS:"
110 INPUT R$(1,1)
120 IF R$(1,1) THEN 490
130 PRINT LIN(1);"WELCOME TO H/P STADIUM FOR TODAY'S GAME"
140 PRINT "THE GAME WILL BE BETWEEN YOU AND ME."
150 PRINT "SINCE I LIVE HERE, I'M THE HOME TEAM!!"
160 PRINT "IN A MINUTE, YOU WILL BE GIVEN YOUR "
170 PRINT "TEAM'S STARTING ROSTER FROM WHICH YOU"
180 PRINT "WILL SELECT A STARTING LINEUP."
190 PRINT "WHEN ASKED TO, SIMPLY INPUT THE # OF"
200 PRINT "YOUR CHOICES";LIN(1);"FOR COMMAND LIST, TYPE '6'"
210 PRINT "FOR COMMAND CHOICE." NOTE: JUST HITTING"
220 PRINT "RETURN AFTER 'COMMAND' WILL PITCH"
230 PRINT "TO THE BATTER. (HIT RETURN TO CONTINUE)"
240 ENTER 255,E,RS
250 PRINT LIN(1);"THE LIST OF COMMANDS IS QUITE SELF-"
260 PRINT "EXPLANATORY, EXCEPT THAT #4 GIVES A"
270 PRINT "LIST OF BASEBALL POSITIONS WITH THEIR"
280 PRINT "CORRESPONDING #; AND FOR #10, TYPE '99'"
290 PRINT "AND YOU WILL RECEIVE A LIST OF ALL"
300 PRINT "STRATEGIES AVAILABLE TO YOU"
310 PRINT LIN(1);"STRATEGY #17 (PLAYER SUBSTITUTION) IS"
320 PRINT "TO EITHER CHANGE PITCHERS OR PINCH"
330 PRINT "HIT FOR A BATTER CURRENTLY UP OR"
340 PRINT "REPLACE ONE WHILE IN THE FIELD"
350 PRINT "BUT YOU CAN ONLY SUBSTITUTE FOR A"
360 PRINT "POSITION ONCE PER GAME (TWICE FOR"
370 PRINT "PITCHERS).";
380 ENTER 255,E,RS
390 PRINT LIN(1);"IN BOTH COMMAND #1 AND STRATEGY #17,"
400 PRINT "A FIELDING # AND RUNNING # APPEAR."
410 PRINT "THESE RATE EACH PLAYER'S ABILITIES "
420 PRINT "WITH EITHER A 1 OR 2. #17 CHANGES THEM"
430 PRINT "ALSO."
440 PRINT "TO TAKE THE HUNT OFF AFTER EXECUTING"
450 PRINT "IT, JUST REPEAT THE STRATEGY #."
460 PRINT "GOOD LUCK AND MAY THE BEST"
470 PRINT "MACHINE WIN!!";LIN(3)
480 ENTER 255,E,RS
490 C=6
500 MAT R=ZER(4)
510 MAT S=ZER(4)
520 MAT T=ZER(2)
530 MAT I=ZER(2,10)
540 MAT L=ZER(2)
550 MAT E=ZER(2)
560 MAT A=ZER

```

```

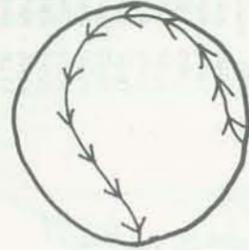
570 MAT H=ZER(2)
580 P=PO=H=I=S=59=K=K1=K2=K3=S=B=D7=L=0
590 L=FNA(3)--2
600 GOTO 810
610 PRINT LIN(1);"INPUT YOUR 9 MAN BATTING ORDER BY #:"
620 MAT INPUT CL(9)
630 FOR B=1 TO 9
640 IF CL(B)<1 OR CL(B)>9 OR CL(B)#INT(CL(B)) THEN 790
650 FOR B=B+1 TO 9
660 IF CL(B)=CL(B) THEN 760
670 NEXT B
680 NEXT B
690 FOR J=1 TO 9
700 ALJ(9)=CL(J)
710 ALJ(8)=Y(CCL(J))
720 ALJ(7,9)=C(CJ)
730 ALJ(7,8)=X(CL(J))
740 NEXT J
750 GOTO 1040
760 PRINT "NO DUPLICATES!"
770 MAT PRINT USING "XDD"IC
780 GOTO 610
790 PRINT "ILLEGAL #:"
800 GOTO 770
810 FOR A=1 TO 18
820 FOR A=6 TO 7
830 ALA(AOJ)=FNA(5) MIN 2
840 NEXT AO
850 N=(FNA(120)+215)/1000
860 IF A=1 OR A=10 THEN 930
870 IF A<10 THEN 900
880 XIA(9)=M
890 GOTO 910
900 YIAJ=M
910 NEXT A
920 GOTO 950
930 N=(FNA(95)+60)/1000
940 GOTO (A=10)+1 OF 900,880
950 PRINT "HERE IS YOUR STARTING TEAM:";LIN(1)
960 PRINT "POSITION", " # AVG."
970 PRINT "-----", " -"
980 FOR F2=1 TO 9
990 READ L$
1000 PRINT L$,F2;
1010 PRINT USING ".3D";Y(F2)
1020 NEXT F2
1030 GOTO 610
1040 FOR I=1 TO 9
1050 FOR TO=1 TO 2
1060 I4=INT((1-1)/10)
1070 I5=INT((1/10)-INT(I4/10))*10
1080 I5=10*(I5+0)+I5
1090 PRINT LIN(1);"INNING";I;"TEAM"TO
1100 MAT R=ZER
1110 O=B=0
1120 F=S=B=K1=K2=K3=H2=R2=P0=0
1130 LLTOJ=LLTOJ+1-(9*(LLTOJ >= 9))
1140 IF P THEN 6690
1150 PRINT LIN(1);"BATTER UP! #";LLTOJ;
1160 ALFN(B(LLTOJ),2)=FNC(2)+1
1170 RESTORE
1180 FOR K=1 TO CL(LLTOJ)
1190 READ L$
1200 NEXT K
1210 PRINT LIN(1);L$ AVG.:";
1220 PRINT USING ".3D";FNC(8)
1230 GOTO 4910
1240 GOSUB 3800
1250 PRINT LIN(1);"COMMAND:";
1260 S9=D7=K2=F1=L=F=0

```

```

1270 ENTER 255,L7,C
1280 PRINT
1290 GOTO C+1 OF 1330,3060,3220,3380,3470,4040,4100,1240,4270,5370,5470
1300 PRINT "FOR LIST OF COMMANDS, TYPE 6"
1310 C=6
1320 GOTO 1250
1330 M=RND(O)*(.1*(FNA(3)+1))+((FNA(10)<5)*(#-1)*(W/25))+(.05*(FNA(2)=1)
1340 IF M <= FNC(8) THEN 2520
1350 GOTO FNA(2)*K3 OF 6090
1360 GOTO INT(M*20) OF 1380,1650,2790,1510,1580,1510,1580,2370,1640,1380
1370 GOTO INT(M*20)-10 OF 2790,2790,1510,1510,1390,1380,1580,1580,1330,1650
1380 GOTO FNA(3) OF 1640
1390 PRINT "FOUL BALL ";
1400 F1=1
1410 IF FNA(4)#4 THEN 1450
1420 PRINT "CAUGHT...OUT!!"
1430 GOSUB 6730
1440 GOTO 1630
1450 S=S+1 MIN 2+K3
1460 IF S=3 THEN 1610
1470 PRINT USING 1480;B,S
1480 IMAGE "(,,"D,","-","D,")"
1490 IF H2 AND NOT F1 THEN 5960
1500 GOTO 1250
1510 PRINT LIN(1);"BALL ";
1520 B=B+1
1530 IF B<4 THEN 1470
1540 PRINT "4--WALK"
1550 ALFNB(L(TO)),2)=FNC(2)-1
1560 GOSUB 5210
1570 GOTO 1120
1580 PRINT LIN(1);"STRIKE ";
1590 S=S+1
1600 IF S<3 THEN 1470
1610 PRINT "3--OUT!!"
1620 GOSUB 6730
1630 GOTO 2170
1640 GOTO FNA(2) OF 2790
1650 PRINT "GROUND BALL TO ";
1660 E=FNA(5)
1670 E=E*(E#1)
1680 RESTORE
1690 FOR M2=1 TO E
1700 READ L2
1710 NEXT M2
1720 PRINT "THE ";L2
1730 GOTO 1960
1740 PRINT "THROW TO 2ND BASE...";
1750 IF E#4 THEN 1770
1760 PRINT "SHORTSTOP COVERING...";
1770 H=H1=D7=1
1780 O=0+1
1790 GOSUB 6730
1800 PRINT "OUT"
1810 R(1)=0
1820 IF O=3 OR FNA(3)#1 THEN 1890
1830 PRINT "THROW TO 1ST BASE...";
1840 GOSUB 6730
1850 O=0+1
1860 H=0
1870 IF NOT D7 THEN 1890
1880 PRINT "DOUBLE PLAY!! ";
1890 GOSUB (O-3)+1 OF 4430
1900 GOTO 2180
1910 IF O=0 AND W=1 AND ALFNB(R(1)),7)=1 AND FNA(10)=1 THEN 1930
1920 GOTO 1740
1930 PRINT "WOW!!! TRIPLE PLAY!!!";
1940 O=3
1950 GOTO 2180

```



```

2650 PRINT "DOUBLE...";
2660 H=2
2670 H1=H*(FNA(3)+K2)=1)+H2
2680 GOTO 2100
2690 PRINT "TRIPLE!! ";
2700 H=H1=3
2710 GOTO 2100
2720 PRINT "HOME RUN!!!"
2730 R2=1
2740 H=0
2750 ALFNB(L(TO)),5)=FNC(5)+1
2760 H1=3
2770 GOSUB 4450
2780 GOTO 1120
2790 E=FNA(9)
2800 IF K3 THEN 1650
2810 IF FNA(4)=1 AND E#2 THEN 2870
2820 IF E>6 THEN 2850
2830 PRINT "POP UP TO ";
2840 GOTO 2890
2850 PRINT "FLY BALL TO ";
2860 GOTO 2890
2870 PRINT "LINE DRIVE AT ";
2880 L=1
2890 F=1
2900 GOSUB 6730
2910 RESTORE
2920 FOR M2=1 TO E
2930 READ L2
2940 NEXT M2
2950 PRINT "THE ";L2
2960 IF RND(O)-1-(E/85) OR E<7 THEN 3010
2970 EL3-TOJ=EL3-TOJ+1
2980 PRINT "ERROR...BATTER IS SAFE ON 2ND!!"
2990 H=H1=2
3000 GOTO 2100
3010 IF E>6 AND O<2 AND FNA(5)<3 AND B0 AND NOT L THEN 3030
3020 GOTO 2170
3030 PRINT "OUT, BUT THE RUNNER(S) TAG UP AND MOVE UP ";
3040 GOSUB 4430
3050 GOTO 2170
3060 REM *PLAYER STATS*
3070 PRINT "WHICH TEAM (1=YOURS 2=MINE)";
3080 INPUT D0
3090 IF NOT D0 THEN 1250
3100 IF D0#1 AND D0#2 THEN 3070
3110 PRINT "WHAT IS HIS BATTING POS. #";
3120 INPUT D1
3130 IF NOT D1 THEN 1250
3140 A2=" AB H RBIR F# R#"
3150 FOR C6=2 TO 7
3160 PRINT USING 3170;A2;C6*3-2,C6*3),A(1)+(9*(D0=2)),C6J
3170 IMAGE 3A,"",2D
3180 NEXT C6
3190 PRINT "AVG. ";
3200 PRINT USING ".3D";A(1)+(9*(D0=2)),8J
3210 GOTO 1250
3220 PRINT "TIME IS CALLED...";
3230 GOSUB FNA(5) OF 3280,3300,3320,3340,3360
3240 ENTER 200,W,R5
3250 PRINT LIN(2)
3260 C=0
3270 GOTO 1250
3280 PRINT "THE PITCHER HAS TO GO TO THE BATHROOM";
3290 RETURN
3300 PRINT "THE FAN FELL ASLEEP.";
3310 RETURN
3320 PRINT "THE RIGHT FIELDER HAS TO TIE HIS SHOE.";
3330 RETURN

```



```

1960 R=S*RND(O)*A(E*(9*(TO=1)),7)+H1-K3
1970 IF R<2.5 THEN 2290
1980 GOTO F OF 2160
1990 IF R(1) AND R(2) THEN 1910
2000 IF R(1) THEN 1740
2010 PRINT "THROW TO 1ST BASE...";
2020 IF E#3 AND NOT D7 THEN 2040
2030 PRINT "PITCHER COVERING...";
2040 IF RND(O)-1-(FNC(6)+2.5)/100 THEN 2120
2050 PRINT "RUNNER BEATS IT OUT!!"
2060 H(TO)=H(TO)+1
2070 ALFNB(L(TO)),3)=FNC(3)+1
2080 ALFNB(L(TO)),8)=FNC(8)+FNA(8)/1000) MIN .37
2090 H=1
2100 GOSUB 4430
2110 GOTO 1120
2120 H=0
2130 H1=1
2140 PRINT "OUT"
2150 IF O>1 OR B0=0 THEN 2170
2160 GOSUB 4430
2170 O=0+1
2180 PRINT USING 2190;O
2190 IMAGE D," OUT"
2200 IF O<3 THEN 1120
2210 GOSUB 3800
2220 IF I#7 OR TO#1 THEN 2260
2230 PRINT "7TH INNING STRETCH TIME...";
2240 ENTER 200,U4,H#
2250 PRINT LIN(2)
2260 NEXT I
2270 NEXT I
2280 GOTO 3580
2290 PRINT "THE BALL IS BOBBLED...";
2300 IF RND(O)-1-(FNC(6)+E/100)/10 THEN 1990
2310 PRINT "RUNNER IS SAFE ON 1ST ON THE ERROR!!"
2320 H=1
2330 H1=1*(FNA(2)=K2)
2340 EL3-TOJ=EL3-TOJ+1
2350 GOSUB 4430
2360 GOTO 1120
2370 GOTO FNA(6) OF 1330,2380,2420,1330,1330,1330
2380 PRINT "BATTER IS HIT BY THE PITCH!!"
2390 ALFNB(L(TO)),2)=FNC(2)-1
2400 GOSUB 5210
2410 GOTO 1120
2420 IF R(1)=R(2) AND R(2)=R(3) THEN 1510
2430 GOTO FNA(3) OF 2440,2470,2490
2440 PRINT "WILD PITCH"
2450 GOSUB 4430
2460 GOTO 1510
2470 PRINT "PASSED BALL"
2480 GOTO 2450
2490 PRINT "BALK"
2500 GOSUB 4430
2510 GOTO 1250
2520 REM *HIT*
2530 IF K3 THEN 6090
2540 PRINT "BASE HIT!"
2550 ALFNB(L(TO)),8)=FNC(8)+FNA(8)/1000) MIN .37
2560 ALFNB(L(TO)),3)=FNC(3)+1
2570 H(TO)=H(TO)+1
2580 M1=FNA(20)-((FNA(8)+C(1)(TOJ)=1)))
2590 IF M1>11 AND M1<16 THEN 2650
2600 GOTO M1-18 OF 2690,2720
2610 PRINT "SINGLE...";
2620 H=1
2630 H1=((FNA(2)-K1) MAX 1)+H2-K2
2640 GOTO 2100

```

```

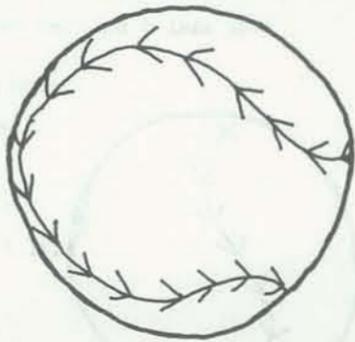
3340 PRINT "TO DUST OFF THE HOME PLATE UMPIRE";
3350 RETURN
3360 PRINT "I'M BORED.";
3370 RETURN
3380 PRINT "HERE IS YOUR LINE-UP:"
3390 PRINT "POS.#","AVG."
3400 PRINT "-----"
3410 FOR I=1 TO 9
3420 PRINT C(I),
3430 NEXT I
3440 PRINT USING ".3D";A(6),8J
3450 C=0
3460 GOTO 1250
3470 PRINT " #","POSITION"
3480 PRINT "-----"
3490 DATA "PITCHER","CATCHER","1ST BASEMAN","2ND BASEMAN","3RD BASEMAN"
3500 DATA "SHORTSTOP","LEFT FIELDER","CENTER FIELDER","RIGHT FIELDER"
3510 RESTORE
3520 FOR D=1 TO 9
3530 READ L2
3540 PRINT D,L2
3550 NEXT D
3560 C=0
3570 GOTO 1250
3580 IF T(1)=T(2) THEN 3710
3590 PRINT "GAME OVER!!"
3600 GOTO (T(1)-T(2)) OF 3640
3610 IF T(1)=T(2) THEN 3650
3620 PRINT "I WIN (AS EXPECTED)";
3630 GOTO 3650
3640 PRINT "YOU WIN";
3650 IMAGE DD,"-",DD
3660 GOSUB (O#3) OF 3800
3670 PRINT LIN(1);"FINAL SCORE:";
3680 PRINT USING 3650;T(1) MAX T(2),T(1) MIN T(2)
3690 STOP
3700 GOSUB 3800
3710 IF (I-1)/10=INT((I-1)/10) THEN 3740
3720 FOR I=1 TO 1
3730 GOTO 1050
3740 PRINT "THIS IS THE LAST TIME THIS SCOREBOARD"
3750 PRINT "WILL BE SHOWN. THE INNING COUNTERS WILL"
3760 PRINT "BE CLEARED, BUT NOT THE SCORE."
3770 MAT 1=ZER
3780 GOTO 3720
3790 NEXT I
3800 REM SCOREBOARD SUBROUTINE
3810 I=1-(INT(I/10))*10
3820 I=I+1*(I<10)
3830 PRINT
3840 FOR I=1 TO 10
3850 PRINT USING "#,DDX";(I0+I4*10)
3860 NEXT I0
3870 PRINT " R H E"
3880 FOR U=1 TO 13
3890 PRINT "----";
3900 NEXT U
3910 PRINT
3920 FOR Y=1 TO 2
3930 FOR O=1 TO 16-((TO=1)*(Y=2))
3940 PRINT USING "#,DDX";I(Y),OJ
3950 NEXT O
3960 PRINT SPA(30-(16-(TO=1)*(Y=2)*3)-(I6*2));
3970 PRINT USING "#,DDX,DDX,DD";T(Y),H(Y),E(Y)
3980 PRINT
3990 NEXT Y
4000 IF T(1)>T(2) AND TO=1 AND I=9 THEN 3590
4010 C=0
4020 RETURN

```

```

4030 RETURN
4040 REM*EARLY END TO THE GAME*
4050 PRINT "END THE GAME EARLY. ARE YOU SURE";
4060 INPUT S$(1,1)
4070 IF S$="Y" THEN 1250
4080 PRINT "GAME CALLED DUE TO RAIN."
4090 GOTO 3590
4100 REM *COMMAND LIST*
4110 PRINT LIN(1);"COMMANDS:"
4120 PRINT "-----"
4130 PRINT "0--PITCH TO THE BATTER-*"
4140 PRINT "1--STATS ON ANY PLAYER"
4150 PRINT "2--CALLS A TIME-OUT"
4160 PRINT "3--GIVES YOUR LINE-UP"
4170 PRINT "4--POSITIONS BY #S"
4180 PRINT "5--ENDS GAME EARLY."
4190 PRINT "6--GIVES THIS LIST OF COMMANDS"
4200 PRINT "7--PRINTS SCOREBOARD"
4210 PRINT "8--GIVES STATUS REPORT"
4220 PRINT "9--GIVES COMPUTER'S LINE-UP"
4230 PRINT "10-INPUTTING YOUR STRATEGIES"
4240 C=0
4250 PRINT LIN(1);"*-OR JUST HIT RETURN"
4260 GOTO 1250
4270 REM *STATUS REPORT*
4280 PRINT "OUTS: "
4290 PRINT "INNING:"
4300 PRINT "SCORE:";
4310 PRINT USING 3650;T(1) MAX T(2),T(1) MIN T(2)
4320 PRINT "TEAM PRESENTLY AT BAT: ";
4330 GOTO TO OF 4360
4340 PRINT "ME"
4350 GOTO 4370
4360 PRINT "YOU"
4370 PRINT "BATTER UP IS #"

```



```

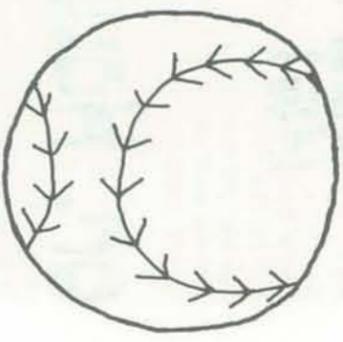
4720 RETURN
4730 PRINT B$(V*3-2,V*3);
4740 B0=B0+1
4750 IF R((V+1) MIN 3) OR R((V+2) MIN 3) THEN 4770
4760 GOTO 4700
4770 IF V=3 THEN 4790
4780 PRINT " AND ";
4790 GOTO 4700
4800 PRINT "BASES ";
4810 H=0
4820 IF B2 THEN 4860
4830 PRINT "LOADED"
4840 B0=3
4850 RETURN
4860 PRINT "EMPTY"
4870 RETURN
4880 R2=R2+1
4890 A(FNB(R(A)),5)=A(FNB(R(A)),5)+1
4900 GOTO 4530
4910 REM *COMPUTER STRATEGIES*
4920 A0=FNC(8)
4930 A1=A(FNB(L(1))+1-(9*(L(1)=9))),8;
4940 GOTO TO OF 4950,4990
4950 IF R(2) AND NOT R(1) AND 0<2 AND ABS(T(1)-T(2))<3 AND A0>A1+.15 AND
A0>.28 THEN 5030
4960 IF R(1) AND 0<2 THEN 5070
4970 IF R(3) AND ABS(T(1)-T(2))<2 THEN 5100
4980 GOTO 1250
4990 IF R(1) AND FNA(5)=1 AND NOT R(2) AND ABS(T(1)-T(2))<2 THEN 5190
5000 IF B0=2 AND ABS(T(1)-T(2))<3 AND A0>.28 AND FNA(3)=1 THEN 5130
5010 IF A0<.21 AND FNA(5)<3 AND B0=1 THEN 5160
5020 GOTO 1250
5030 PRINT "YOUR MAN IS BEING INTENTIONALLY WALKED."
5040 A(FNB(L(1)),2)=FNC(2)-1
5050 GOSUB 5210
5060 GOTO 1120
5070 PRINT "MY INFIELD IS AT D.P. DEPTH"
5080 K1=1
5090 GOTO 5180
5100 PRINT "MY INFIELD IS IN TO GUARD AGAINST THE RUN."
5110 K2=1
5120 GOTO 5180
5130 PRINT "HIT AND RUN IS ON"
5140 H2=1
5150 GOTO 5180
5160 PRINT "BATTER IS PREPARED TO BUNT"
5170 K3=1
5180 GOTO 1250
5190 K=2
5200 GOTO 5690
5210 REM *ADVANCE IF FORCED*
5220 R2=0
5230 MAT S=ZER
5240 S(1)=L(1)
5250 IF NOT R(1) THEN 5340
5260 S(2)=R(1)
5270 IF NOT R(2) THEN 5350
5280 S(3)=R(2)
5290 IF NOT R(3) THEN 5320
5300 R2=1
5310 S(4)=R(3)
5320 MAT R=S
5330 GOTO 4550
5340 S(2)=R(2)
5350 S(3)=R(3)
5360 GOTO 5320
5370 REM *COMPUTER LINEUP*
5380 PRINT "HERE IS MY LINE-UP:"
5390 PRINT "POS. #","AVG."
5400 PRINT "-----","-----"

```

```

5410 FOR Q3=1 TO 9
5420 PRINT A(Q3+9,9);
5430 PRINT USING ".3D";A(Q3+9,8)
5440 NEXT Q3
5450 C=0
5460 GOTO 1250
5470 REM *PLAYER'S STRATEGIES*
5480 PRINT LIN(1);"YOUR STRATEGY #";
5490 R2=C=0
5500 INPUT C0
5510 IF C0=99 THEN 6260
5520 IF NOT C0 THEN 1250
5530 IF C0<11 OR C0>17 OR C0#INT(C0) THEN 5550
5540 GOTO C0-10 OF 5590,5830,5870,5630,5910,6000,6410
5550 PRINT "ILLEGAL #...TYPE 99 FOR LIST OF STRATEGIES"
5560 GOTO 5480
5570 PRINT "ILLEGAL AT THIS TIME."
5580 GOTO 5480
5590 REM *INTENTIONAL WALK*
5600 IF TO#2 THEN 5570
5610 PRINT "THANK YOU FOR THE INTENTIONAL WALK"
5620 GOTO 5040
5630 REM *ATTEMPTED STEAL*
5640 IF TO#1 AND B0=0 THEN 5570
5650 PRINT "WHAT BASE DO YOU WANT TO STEAL";
5660 INPUT K
5670 IF K<2 OR K>4 OR INT(K)#K THEN 5570
5680 IF R(K-1)=0 OR R(K) THEN 5570
5690 PRINT "RUNNER GOES!!!!";LIN(1)
5700 IF FNA(K*2)>(A(FNB(R(K-1)),6)*1.75) THEN 5780
5710 R2=R2+(K=4)
5720 PRINT "SAFE"
5730 R(K)=R(K-1)
5740 R(K-1)=0
5750 GOSUB (0#3) OF 4550
5760 H2=0
5770 GOTO (0#3)+1 OF 2200,1250
5780 PRINT "OUT"
5790 O=O+1
5800 PRINT "OUT"
5810 GOTO 5740
5820 REM *INFIELD ADJUSTMENTS*
5830 IF TO#2 OR K1 OR K2 OR NOT R(1) OR O>1 THEN 5570
5840 PRINT "YOUR INFIELD IS AT D.P. DEPTH";LIN(1)
5850 K1=1
5860 GOTO 1250
5870 IF TO#2 OR K1 OR K2 THEN 5570
5880 PRINT "YOUR INFIELD IS NOW IN TO GUARD AGAINST THE RUN";LIN(1)
5890 K2=1
5900 GOTO 1250
5910 REM *HIT AND RUN*
5920 IF TO#1 OR B0=0 THEN 5570
5930 PRINT LIN(1);"HIT AND RUN ON FOR BATTER PRESENTLY UP";LIN(1)
5940 H2=1
5950 GOTO 1250
5960 PRINT "(DUE TO HIT AND RUN) LEAD ";
5970 FOR K=4 TO 2 STEP -1
5980 IF R(K-1) THEN 5690
5990 NEXT K
6000 REM *BUNT*
6010 IF TO#1 THEN 5570
6020 IF K3 THEN 6060
6030 K3=1
6040 PRINT LIN(1);"BATTER WILL BUNT";LIN(1)
6050 GOTO 5180
6060 PRINT LIN(1);"BUNT IS OFF";LIN(1)
6070 K3=0
6080 GOTO 1250
6090 PRINT "BALL BUNTED AT...";

```



```

6100 GOTO FNA(FNC(6)+FNC(7)) OF 6170,6110,6130,6150
6110 PRINT "AND MISSED."
6120 GOTO 1580
6130 PRINT "AND GOES FOUL...FOUL BALL";LIN(1);"STRIKE ";
6140 GOTO 1450
6150 PRINT "AND CAUGHT...OUT"
6160 GOTO 1620
6170 PRINT "SUCCESSFULLY...";
6180 GOTO FNA(3) OF 6240,2050
6190 PRINT "THROW TO 1ST BASE"
6200 IF FNA(1+(FNC(7))>2) THEN 2050
6210 PRINT "OUT"
6220 GOSUB 4430
6230 GOTO 2170
6240 PRINT "CAUGHT BY THE CATCHER...";
6250 GOTO 2170
6260 PRINT "STRATEGIES";LIN(1);"-----"
6270 GOTO TO OF 6330
6280 PRINT SPA(5);"***FIELDING STRATEGIES"
6290 PRINT "11--INTENTIONAL WALK"
6300 PRINT "12--MOVE INFIELD TO DP DEPTH"
6310 PRINT "13--MOVE INFIELD IN TO GUARD AGAINST A RUN"
6320 GOTO 6370
6330 PRINT SPA(5);"***HITTING STRATEGIES"
6340 PRINT "14--ATTEMPT A STOLEN BASE"
6350 PRINT "15--PUT HIT AND RUN ON"
6360 PRINT "16--LAY DOWN A BUNT"
6370 PRINT SPA(5);"***USABLE AT ANYTIME"
6380 PRINT "0--RETURN TO REGULAR COMMANDS WITHOUT A STRATEGY"
6390 PRINT "17--SUBSTITUTE FOR A PLAYER"
6400 GOTO 5480
6410 REM *PLAYER SUBSTITUTION*
6420 PRINT "WHAT IS THE PLAYERS FIELDING #";
6430 INPUT M9
6440 IF M9<1 OR M9>9 OR M9#INT(M9) THEN 5570
6450 IF M9=1 THEN 6750
6460 IF A(M9,1)=1 THEN 5570
6470 A(M9,1)=1
6480 FOR M8=2 TO 8
6490 A(M9,M8)=0
6500 NEXT M8
6510 A(M9,6)=FNA(5) MIN 2
6520 A(M9,7)=FNA(5) MIN 2
6530 IF M9=1 THEN 6620
6540 A(M9,8)=(FNA(167)+167)/1000
6550 PRINT "NEW AVGS.:"
6560 PRINT "FIELDING #";A(M9,6)
6570 PRINT "RUNNING #";A(M9,7)
6580 PRINT "BATTING AVG.:"
6590 PRINT USING ".3D";A(M9,8)
6600 IF NOT PO THEN 1150
6610 GOTO (FNB(L(1))=M9)+1 OF 1250,1150
6620 A(M9,8)=(FNA(9)+6)/100
6630 W=FNA(3)-2
6640 IF C(L(1))#1 THEN 6550
6650 A(1,8)=(FNA(133)+200)/1000
6660 PRINT "PINCH HITTER'S AVG:"A(1,8)
6670 P=PO=1
6680 GOTO 1250
6690 A(1,8)=(FNA(90)+67)/1000
6700 PRINT "NEW PITCHER'S AVGS:"
6710 P=0
6720 GOTO 6560
6730 A(FNB(L(1)),8)=(FNC(8)-FNA(8)/1000) MAX .065
6740 RETURN
6750 A(M9,M9)=A(M9,M9)+.5
6760 IF A(M9,M9)>1 THEN 5570
6770 GOTO 6480
6780 END

```


Newett Awl and the Goat

BY GORDON FRENCH

PART I: THE PROBLEM

The following is the first part of a letter from Gordon to Robert Reiling, editor of the Homebrew Computer Club Newsletter. The Newsletter is distributed free at club meetings and is available (upon receipt of your reasonable donation) from P.O. Box 626, Mountain View, CA 94042.

The second part of the letter will be printed next issue along with a sample program solution. But the challenge goes out to all the readers - write your own program that solves the problem! Send in a listing and a sample run. The early postmarks will get mentioned next issue. - RZ

Dear Bob,
We were talking about Newett Awl the other day, and I forgot to tell you what happened the night that he showed up with a fistful of software for the library.

He had no sooner arrived than there was another knock at the door. It was my neighbor asking if Newett would move his microbus from in front of his driveway. While Newett moved his car (in front of my driveway), my neighbor asked me what we were doing. I know that my neighbor is a part-time cop, and since Newett looks so shady, I thought I'd better explain. I began to tell him how good Newett was at solving computer problems and could we figure something out for him. Newett scratched himself and asked what the problem was.

My neighbor said if you had a circular lawn 20 feet in diameter and wanted to stake a goat at the periphery so that the goat would eat half the grass, how long would the rope be?

Newett asked if he could borrow a pencil and began to make a little sketch. It looked like:

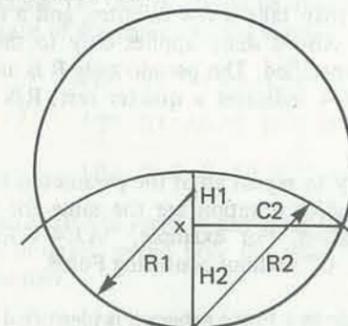


Figure 1

Then he wandered off into the computer room while I made some coffee and excuses for my dog burying a bone in my neighbor's tulips. Pretty soon, Newett came out carrying a piece of paper. He handed the paper to my neighbor. It said:

```

RUN
DIAMETER OF LAWN = 20
PERCENTAGE OF LAWN GOAT IS TO EAT = 50
AREA OF LAWN IS 314.15927 SQUARE UNITS
GOAT'S PART IS 157.07964 SQUARE UNITS
CALCULATING . . .
MAKE THE ROPE 11.587285 UNITS LONG
READY
    
```

My neighbor looked for a moment like he was going to flash his badge and frisk Newett so I quickly asked Newett how he had done it.

Newett mumbled something about the problem not being ideally suited to being solved digitally. He said that it was really a problem for integral calculus, but for small percentages of the lawn, you could solve the problem by considering the goat's area as two circular segments. He said that the formula for calculating the area of a circular segment is: $A = 1/2 [RL - C(R - H)]$ where R is the radius, L is the length of the arc, and C is the base. Diagrammed it looks like:

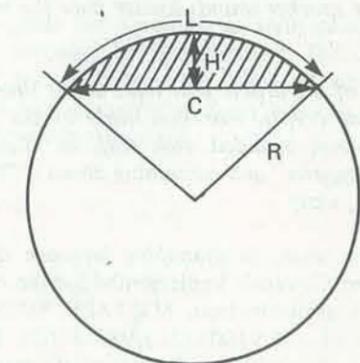


Figure 2

What did Newett's program look like? What happens to Newett at the end of the letter? What about his microbus? For answers to these questions and for the names of the people who send in solutions, look for Newett's Goat in the next issue. □

new PETs. Line 4 in my program when LISTed will function properly only on the 8K PET. To be used on the new PETs, it must be changed to the following:

```

4 POKE158,5:PRINT"[CLR] [2DOWN]":
FORX=1TO4:POKE622+X,13:
PRINTX:NEXTX:PRINT"[HOME]":END
    
```

These changes are due to the keyboard buffer and its counter locations changing.

To implement the perpetual feature on the 8K PET, use LINE 9 as listed. On the NEW 16K and 32K PETs change the POKEs in line 9 to the following:

```
POKE 158,1:POKE 623,13
```

If you would like to be mean, you can disable the STOP key so that the only way to stop the LIST would be to turn off the PET. To do this you must insert one POKE just before the first POKE in LINE 9. For the old 8K PET insert:

```
POKE537,136:
```

For the NEW 32K PET insert:

```
POKE144,46:
```

For the NEW 16K PET insert:

```
POKE144,PEEK(144)+3:
```

I do not have a 16K PET so cannot check what exact value to POKE. It should be three more than what is there to start with. Thus you could do a:

```
?PEEK(144)
```

and add three to the result and use that number in your POKE.

PLEASE WRITE

More fun next time. Please write with your comments, ideas, and suggestions as well as your entry program on tape for my CHALLENGE. Write to: Len Lindsay, Editor, *PET Gazette*, 1929 Northport Drive, Room 6, Madison, WI 53704. □

Listing of Perpetual Animated List (With special keys as they appear on PET screen.)

```

READY.
1 FOR X=1024TO3000:IF PEEK(X)=255THENPO
K=1:PEEK(X)=94THENPOKE X,17
2 PEEK(X)=94THENPOKE X,17
3 PEEK(X)=94THENPOKE X,17
4 PEEK(X)=94THENPOKE X,17
5 PEEK(X)=94THENPOKE X,17
6 PEEK(X)=94THENPOKE X,17
7 PEEK(X)=94THENPOKE X,17
8 PEEK(X)=94THENPOKE X,17
9 PEEK(X)=94THENPOKE X,17
10 PEEK(X)=94THENPOKE X,17
11 PEEK(X)=94THENPOKE X,17
12 PEEK(X)=94THENPOKE X,17
13 PEEK(X)=94THENPOKE X,17
14 PEEK(X)=94THENPOKE X,17
15 PEEK(X)=94THENPOKE X,17
16 PEEK(X)=94THENPOKE X,17
17 PEEK(X)=94THENPOKE X,17
18 PEEK(X)=94THENPOKE X,17
19 PEEK(X)=94THENPOKE X,17
20 PEEK(X)=94THENPOKE X,17
21 PEEK(X)=94THENPOKE X,17
22 PEEK(X)=94THENPOKE X,17
23 PEEK(X)=94THENPOKE X,17
24 PEEK(X)=94THENPOKE X,17
25 PEEK(X)=94THENPOKE X,17
26 PEEK(X)=94THENPOKE X,17
27 PEEK(X)=94THENPOKE X,17
28 PEEK(X)=94THENPOKE X,17
29 PEEK(X)=94THENPOKE X,17
30 PEEK(X)=94THENPOKE X,17
31 PEEK(X)=94THENPOKE X,17
32 PEEK(X)=94THENPOKE X,17
33 PEEK(X)=94THENPOKE X,17
34 PEEK(X)=94THENPOKE X,17
35 PEEK(X)=94THENPOKE X,17
36 PEEK(X)=94THENPOKE X,17
37 PEEK(X)=94THENPOKE X,17
38 PEEK(X)=94THENPOKE X,17
39 PEEK(X)=94THENPOKE X,17
40 PEEK(X)=94THENPOKE X,17
41 PEEK(X)=94THENPOKE X,17
42 PEEK(X)=94THENPOKE X,17
43 PEEK(X)=94THENPOKE X,17
44 PEEK(X)=94THENPOKE X,17
45 PEEK(X)=94THENPOKE X,17
46 PEEK(X)=94THENPOKE X,17
47 PEEK(X)=94THENPOKE X,17
48 PEEK(X)=94THENPOKE X,17
49 PEEK(X)=94THENPOKE X,17
50 PEEK(X)=94THENPOKE X,17
51 PEEK(X)=94THENPOKE X,17
52 PEEK(X)=94THENPOKE X,17
53 PEEK(X)=94THENPOKE X,17
54 PEEK(X)=94THENPOKE X,17
55 PEEK(X)=94THENPOKE X,17
56 PEEK(X)=94THENPOKE X,17
57 PEEK(X)=94THENPOKE X,17
58 PEEK(X)=94THENPOKE X,17
59 PEEK(X)=94THENPOKE X,17
60 PEEK(X)=94THENPOKE X,17
61 PEEK(X)=94THENPOKE X,17
62 PEEK(X)=94THENPOKE X,17
63 PEEK(X)=94THENPOKE X,17
64 PEEK(X)=94THENPOKE X,17
65 PEEK(X)=94THENPOKE X,17
66 PEEK(X)=94THENPOKE X,17
67 PEEK(X)=94THENPOKE X,17
68 PEEK(X)=94THENPOKE X,17
69 PEEK(X)=94THENPOKE X,17
70 PEEK(X)=94THENPOKE X,17
71 PEEK(X)=94THENPOKE X,17
72 PEEK(X)=94THENPOKE X,17
73 PEEK(X)=94THENPOKE X,17
74 PEEK(X)=94THENPOKE X,17
75 PEEK(X)=94THENPOKE X,17
76 PEEK(X)=94THENPOKE X,17
77 PEEK(X)=94THENPOKE X,17
78 PEEK(X)=94THENPOKE X,17
79 PEEK(X)=94THENPOKE X,17
80 PEEK(X)=94THENPOKE X,17
81 PEEK(X)=94THENPOKE X,17
82 PEEK(X)=94THENPOKE X,17
83 PEEK(X)=94THENPOKE X,17
84 PEEK(X)=94THENPOKE X,17
85 PEEK(X)=94THENPOKE X,17
86 PEEK(X)=94THENPOKE X,17
87 PEEK(X)=94THENPOKE X,17
88 PEEK(X)=94THENPOKE X,17
89 PEEK(X)=94THENPOKE X,17
90 PEEK(X)=94THENPOKE X,17
91 PEEK(X)=94THENPOKE X,17
92 PEEK(X)=94THENPOKE X,17
93 PEEK(X)=94THENPOKE X,17
94 PEEK(X)=94THENPOKE X,17
95 PEEK(X)=94THENPOKE X,17
96 PEEK(X)=94THENPOKE X,17
97 PEEK(X)=94THENPOKE X,17
98 PEEK(X)=94THENPOKE X,17
99 PEEK(X)=94THENPOKE X,17
100 PEEK(X)=94THENPOKE X,17
101 PEEK(X)=94THENPOKE X,17
102 PEEK(X)=94THENPOKE X,17
103 PEEK(X)=94THENPOKE X,17
104 PEEK(X)=94THENPOKE X,17
105 PEEK(X)=94THENPOKE X,17
106 PEEK(X)=94THENPOKE X,17
107 PEEK(X)=94THENPOKE X,17
108 PEEK(X)=94THENPOKE X,17
109 PEEK(X)=94THENPOKE X,17
110 PEEK(X)=94THENPOKE X,17
111 PEEK(X)=94THENPOKE X,17
112 PEEK(X)=94THENPOKE X,17
113 PEEK(X)=94THENPOKE X,17
114 PEEK(X)=94THENPOKE X,17
115 PEEK(X)=94THENPOKE X,17
116 PEEK(X)=94THENPOKE X,17
117 PEEK(X)=94THENPOKE X,17
118 PEEK(X)=94THENPOKE X,17
119 PEEK(X)=94THENPOKE X,17
120 PEEK(X)=94THENPOKE X,17
121 PEEK(X)=94THENPOKE X,17
122 PEEK(X)=94THENPOKE X,17
123 PEEK(X)=94THENPOKE X,17
124 PEEK(X)=94THENPOKE X,17
125 PEEK(X)=94THENPOKE X,17
126 PEEK(X)=94THENPOKE X,17
127 PEEK(X)=94THENPOKE X,17
128 PEEK(X)=94THENPOKE X,17
129 PEEK(X)=94THENPOKE X,17
130 PEEK(X)=94THENPOKE X,17
131 PEEK(X)=94THENPOKE X,17
132 PEEK(X)=94THENPOKE X,17
133 PEEK(X)=94THENPOKE X,17
134 PEEK(X)=94THENPOKE X,17
135 PEEK(X)=94THENPOKE X,17
136 PEEK(X)=94THENPOKE X,17
137 PEEK(X)=94THENPOKE X,17
138 PEEK(X)=94THENPOKE X,17
139 PEEK(X)=94THENPOKE X,17
140 PEEK(X)=94THENPOKE X,17
141 PEEK(X)=94THENPOKE X,17
142 PEEK(X)=94THENPOKE X,17
143 PEEK(X)=94THENPOKE X,17
144 PEEK(X)=94THENPOKE X,17
145 PEEK(X)=94THENPOKE X,17
146 PEEK(X)=94THENPOKE X,17
147 PEEK(X)=94THENPOKE X,17
148 PEEK(X)=94THENPOKE X,17
149 PEEK(X)=94THENPOKE X,17
150 PEEK(X)=94THENPOKE X,17
151 PEEK(X)=94THENPOKE X,17
152 PEEK(X)=94THENPOKE X,17
153 PEEK(X)=94THENPOKE X,17
154 PEEK(X)=94THENPOKE X,17
155 PEEK(X)=94THENPOKE X,17
156 PEEK(X)=94THENPOKE X,17
157 PEEK(X)=94THENPOKE X,17
158 PEEK(X)=94THENPOKE X,17
159 PEEK(X)=94THENPOKE X,17
160 PEEK(X)=94THENPOKE X,17
161 PEEK(X)=94THENPOKE X,17
162 PEEK(X)=94THENPOKE X,17
163 PEEK(X)=94THENPOKE X,17
164 PEEK(X)=94THENPOKE X,17
165 PEEK(X)=94THENPOKE X,17
166 PEEK(X)=94THENPOKE X,17
167 PEEK(X)=94THENPOKE X,17
168 PEEK(X)=94THENPOKE X,17
169 PEEK(X)=94THENPOKE X,17
170 PEEK(X)=94THENPOKE X,17
171 PEEK(X)=94THENPOKE X,17
172 PEEK(X)=94THENPOKE X,17
173 PEEK(X)=94THENPOKE X,17
174 PEEK(X)=94THENPOKE X,17
175 PEEK(X)=94THENPOKE X,17
176 PEEK(X)=94THENPOKE X,17
177 PEEK(X)=94THENPOKE X,17
178 PEEK(X)=94THENPOKE X,17
179 PEEK(X)=94THENPOKE X,17
180 PEEK(X)=94THENPOKE X,17
181 PEEK(X)=94THENPOKE X,17
182 PEEK(X)=94THENPOKE X,17
183 PEEK(X)=94THENPOKE X,17
184 PEEK(X)=94THENPOKE X,17
185 PEEK(X)=94THENPOKE X,17
186 PEEK(X)=94THENPOKE X,17
187 PEEK(X)=94THENPOKE X,17
188 PEEK(X)=94THENPOKE X,17
189 PEEK(X)=94THENPOKE X,17
190 PEEK(X)=94THENPOKE X,17
191 PEEK(X)=94THENPOKE X,17
192 PEEK(X)=94THENPOKE X,17
193 PEEK(X)=94THENPOKE X,17
194 PEEK(X)=94THENPOKE X,17
195 PEEK(X)=94THENPOKE X,17
196 PEEK(X)=94THENPOKE X,17
197 PEEK(X)=94THENPOKE X,17
198 PEEK(X)=94THENPOKE X,17
199 PEEK(X)=94THENPOKE X,17
200 PEEK(X)=94THENPOKE X,17
201 PEEK(X)=94THENPOKE X,17
202 PEEK(X)=94THENPOKE X,17
203 PEEK(X)=94THENPOKE X,17
204 PEEK(X)=94THENPOKE X,17
205 PEEK(X)=94THENPOKE X,17
206 PEEK(X)=94THENPOKE X,17
207 PEEK(X)=94THENPOKE X,17
208 PEEK(X)=94THENPOKE X,17
209 PEEK(X)=94THENPOKE X,17
210 PEEK(X)=94THENPOKE X,17
211 PEEK(X)=94THENPOKE X,17
212 PEEK(X)=94THENPOKE X,17
213 PEEK(X)=94THENPOKE X,17
214 PEEK(X)=94THENPOKE X,17
215 PEEK(X)=94THENPOKE X,17
216 PEEK(X)=94THENPOKE X,17
217 PEEK(X)=94THENPOKE X,17
218 PEEK(X)=94THENPOKE X,17
219 PEEK(X)=94THENPOKE X,17
220 PEEK(X)=94THENPOKE X,17
221 PEEK(X)=94THENPOKE X,17
222 PEEK(X)=94THENPOKE X,17
223 PEEK(X)=94THENPOKE X,17
224 PEEK(X)=94THENPOKE X,17
225 PEEK(X)=94THENPOKE X,17
226 PEEK(X)=94THENPOKE X,17
227 PEEK(X)=94THENPOKE X,17
228 PEEK(X)=94THENPOKE X,17
229 PEEK(X)=94THENPOKE X,17
230 PEEK(X)=94THENPOKE X,17
231 PEEK(X)=94THENPOKE X,17
232 PEEK(X)=94THENPOKE X,17
233 PEEK(X)=94THENPOKE X,17
234 PEEK(X)=94THENPOKE X,17
235 PEEK(X)=94THENPOKE X,17
236 PEEK(X)=94THENPOKE X,17
237 PEEK(X)=94THENPOKE X,17
238 PEEK(X)=94THENPOKE X,17
239 PEEK(X)=94THENPOKE X,17
240 PEEK(X)=94THENPOKE X,17
241 PEEK(X)=94THENPOKE X,17
242 PEEK(X)=94THENPOKE X,17
243 PEEK(X)=94THENPOKE X,17
244 PEEK(X)=94THENPOKE X,17
245 PEEK(X)=94THENPOKE X,17
246 PEEK(X)=94THENPOKE X,17
247 PEEK(X)=94THENPOKE X,17
248 PEEK(X)=94THENPOKE X,17
249 PEEK(X)=94THENPOKE X,17
250 PEEK(X)=94THENPOKE X,17
251 PEEK(X)=94THENPOKE X,17
252 PEEK(X)=94THENPOKE X,17
253 PEEK(X)=94THENPOKE X,17
254 PEEK(X)=94THENPOKE X,17
255 PEEK(X)=94THENPOKE X,17
256 PEEK(X)=94THENPOKE X,17
257 PEEK(X)=94THENPOKE X,17
258 PEEK(X)=94THENPOKE X,17
259 PEEK(X)=94THENPOKE X,17
260 PEEK(X)=94THENPOKE X,17
261 PEEK(X)=94THENPOKE X,17
262 PEEK(X)=94THENPOKE X,17
263 PEEK(X)=94THENPOKE X,17
264 PEEK(X)=94THENPOKE X,17
265 PEEK(X)=94THENPOKE X,17
266 PEEK(X)=94THENPOKE X,17
267 PEEK(X)=94THENPOKE X,17
268 PEEK(X)=94THENPOKE X,17
269 PEEK(X)=94THENPOKE X,17
270 PEEK(X)=94THENPOKE X,17
271 PEEK(X)=94THENPOKE X,17
272 PEEK(X)=94THENPOKE X,17
273 PEEK(X)=94THENPOKE X,17
274 PEEK(X)=94THENPOKE X,17
275 PEEK(X)=94THENPOKE X,17
276 PEEK(X)=94THENPOKE X,17
277 PEEK(X)=94THENPOKE X,17
278 PEEK(X)=94THENPOKE X,17
279 PEEK(X)=94THENPOKE X,17
280 PEEK(X)=94THENPOKE X,17
281 PEEK(X)=94THENPOKE X,17
282 PEEK(X)=94THENPOKE X,17
283 PEEK(X)=94THENPOKE X,17
284 PEEK(X)=94THENPOKE X,17
285 PEEK(X)=94THENPOKE X,17
286 PEEK(X)=94THENPOKE X,17
287 PEEK(X)=94THENPOKE X,17
288 PEEK(X)=94THENPOKE X,17
289 PEEK(X)=94THENPOKE X,17
290 PEEK(X)=94THENPOKE X,17
291 PEEK(X)=94THENPOKE X,17
292 PEEK(X)=94THENPOKE X,17
293 PEEK(X)=94THENPOKE X,17
294 PEEK(X)=94THENPOKE X,17
295 PEEK(X)=94THENPOKE X,17
296 PEEK(X)=94THENPOKE X,17
297 PEEK(X)=94THENPOKE X,17
298 PEEK(X)=94THENPOKE X,17
299 PEEK(X)=94THENPOKE X,17
300 PEEK(X)=94THENPOKE X,17
301 PEEK(X)=94THENPOKE X,17
302 PEEK(X)=94THENPOKE X,17
303 PEEK(X)=94THENPOKE X,17
304 PEEK(X)=94THENPOKE X,17
305 PEEK(X)=94THENPOKE X,17
306 PEEK(X)=94THENPOKE X,17
307 PEEK(X)=94THENPOKE X,17
308 PEEK(X)=94THENPOKE X,17
309 PEEK(X)=94THENPOKE X,17
310 PEEK(X)=94THENPOKE X,17
311 PEEK(X)=94THENPOKE X,17
312 PEEK(X)=94THENPOKE X,17
313 PEEK(X)=94THENPOKE X,17
314 PEEK(X)=94THENPOKE X,17
315 PEEK(X)=94THENPOKE X,17
316 PEEK(X)=94THENPOKE X,17
317 PEEK(X)=94THENPOKE X,17
318 PEEK(X)=94THENPOKE X,17
319 PEEK(X)=94THENPOKE X,17
320 PEEK(X)=94THENPOKE X,17
321 PEEK(X)=94THENPOKE X,17
322 PEEK(X)=94THENPOKE X,17
323 PEEK(X)=94THENPOKE X,17
324 PEEK(X)=94THENPOKE X,17
325 PEEK(X)=94THENPOKE X,17
326 PEEK(X)=94THENPOKE X,17
327 PEEK(X)=94THENPOKE X,17
328 PEEK(X)=94THENPOKE X,17
329 PEEK(X)=94THENPOKE X,17
330 PEEK(X)=94THENPOKE X,17
331 PEEK(X)=94THENPOKE X,17
332 PEEK(X)=94THENPOKE X,17
333 PEEK(X)=94THENPOKE X,17
334 PEEK(X)=94THENPOKE X,17
335 PEEK(X)=94THENPOKE X,17
336 PEEK(X)=94THENPOKE X,17
337 PEEK(X)=94THENPOKE X,17
338 PEEK(X)=94THENPOKE X,17
339 PEEK(X)=94THENPOKE X,17
340 PEEK(X)=94THENPOKE X,17
341 PEEK(X)=94THENPOKE X,17
342 PEEK(X)=94THENPOKE X,17
343 PEEK(X)=94THENPOKE X,17
344 PEEK(X)=94THENPOKE X,17
345 PEEK(X)=94THENPOKE X,17
346 PEEK(X)=94THENPOKE X,17
347 PEEK(X)=94THENPOKE X,17
348 PEEK(X)=94THENPOKE X,17
349 PEEK(X)=94THENPOKE X,17
350 PEEK(X)=94THENPOKE X,17
351 PEEK(X)=94THENPOKE X,17
352 PEEK(X)=94THENPOKE X,17
353 PEEK(X)=94THENPOKE X,17
354 PEEK(X)=94THENPOKE X,17
355 PEEK(X)=94THENPOKE X,17
356 PEEK(X)=94THENPOKE X,17
357 PEEK(X)=94THENPOKE X,17
358 PEEK(X)=94THENPOKE X,17
359 PEEK(X)=94THENPOKE X,17
360 PEEK(X)=94THENPOKE X,17
361 PEEK(X)=94THENPOKE X,17
362 PEEK(X)=94THENPOKE X,17
363 PEEK(X)=94THENPOKE X,17
364 PEEK(X)=94THENPOKE X,17
365 PEEK(X)=94THENPOKE X,17
366 PEEK(X)=94THENPOKE X,17
367 PEEK(X)=94THENPOKE X,17
368 PEEK(X)=94THENPOKE X,17
369 PEEK(X)=94THENPOKE X,17
370 PEEK(X)=94THENPOKE X,17
371 PEEK(X)=94THENPOKE X,17
372 PEEK(X)=94THENPOKE X,17
373 PEEK(X)=94THENPOKE X,17
374 PEEK(X)=94THENPOKE X,17
375 PEEK(X)=94THENPOKE X,17
376 PEEK(X)=94THENPOKE X,17
377 PEEK(X)=94THENPOKE X,17
378 PEEK(X)=94THENPOKE X,17
379 PEEK(X)=94THENPOKE X,17
380 PEEK(X)=94THENPOKE X,17
381 PEEK(X)=94THENPOKE X,17
382 PEEK(X)=94THENPOKE X,17
383 PEEK(X)=94THENPOKE X,17
384 PEEK(X)=94THENPOKE X,17
385 PEEK(X)=94THENPOKE X,17
386 PEEK(X)=94THENPOKE X,17
387 PEEK(X)=94THENPOKE X,17
388 PEEK(X)=94THENPOKE X,17
389 PEEK(X)=94THENPOKE X,17
390 PEEK(X)=94THENPOKE X,17
391 PEEK(X)=94THENPOKE X,17
392 PEEK(X)=94THENPOKE X,17
393 PEEK(X)=94THENPOKE X,17
394 PEEK(X)=94THENPOKE X,17
395 PEEK(X)=94THENPOKE X,17
396 PEEK(X)=94THENPOKE X,17
397 PEEK(X)=94THENPOKE X,17
398 PEEK(X)=94THENPOKE X,17
399 PEEK(X)=94THENPOKE X,17
400 PEEK(X)=94THENPOKE X,17
401 PEEK(X)=94THENPOKE X,17
402 PEEK(X)=94THENPOKE X,17
403 PEEK(X)=94THENPOKE X,17
404 PEEK(X)=94THENPOKE X,17
405 PEEK(X)=94THENPOKE X,17
406 PEEK(X)=94THENPOKE X,17
407 PEEK(X)=94THENPOKE X,17
408 PEEK(X)=94THENPOKE X,17
409 PEEK(X)=94THENPOKE X,17
410 PEEK(X)=94THENPOKE X,17
411 PEEK(X)=94THENPOKE X,17
412 PEEK(X)=94THENPOKE X,17
413 PEEK(X)=94THENPOKE X,17
414 PEEK(X)=94THENPOKE X,17
415 PEEK(X)=94THENPOKE X,17
416 PEEK(X)=94THENPOKE X,17
417 PEEK(X)=94THENPOKE X,17
418 PEEK(X)=94THENPOKE X,17
419 PEEK(X)=94THENPOKE X,17
420 PEEK(X)=94THENPOKE X,17
421 PEEK(X)=94THENPOKE X,17
422 PEEK(X)=94THENPOKE X,17
423 PEEK(X)=94THENPOKE X,17
424 PEEK(X)=94THENPOKE X,17
425 PEEK(X)=94THENPOKE X,17
426 PEEK(X)=94THENPOKE X,17
427 PEEK(X)=94THENPOKE X,17
428 PEEK(X)=94THENPOKE X,17
429 PEEK(X)=94THENPOKE X,17
430 PEEK(X)=94THENPOKE X,17
431 PEEK(X)=94THENPOKE X,17
432 PEEK(X)=94THENPOKE X,17
433 PEEK(X)=94THENPOKE X,17
434 PEEK(X)=94THENPOKE X,17
435 PEEK(X)=94THENPOKE X,17
436 PEEK(X)=94THENPOKE X,17
437 PEEK(X)=94THENPOKE X,17
438 PEEK(X)=94THENPOKE X,17
439 PEEK(X)=94THENPOKE X,17
440 PEEK(X)=94THENPOKE X,17
441 PEEK(X)=94THENPOKE X,17
442 PEEK(X)=94THENPOKE X,17
443 PEEK(X)=94THENPOKE X,17
444 PEEK(X)=94THENPOKE X,17
445 PEEK(X)=94THENPOKE X,17
446 PEEK(X)=94THENPOKE X,17
447 PEEK(X)=94THENPOKE X,17
448 PEEK(X)=94THENPOKE X,17
449 PEEK(X)=94THENPOKE X,17
450 PEEK(X)=94THENPOKE X,17
451 PEEK(X)=94THENPOKE X,17
452 PEEK(X)=94THENPOKE X,17
453 PEEK(X)=94THENPOKE X,17
454 PEEK(X)=94THENPOKE X,17
455 PEEK(X)=94THENPOKE X,17
456 PEEK(X)=94THENPOKE X,17
457 PEEK(X)=94THENPOKE X,17
458 PEEK(X)=94THENPOKE X,17
459 PEEK(X)=94THENPOKE X,17
460 PEEK(X)=94THENPOKE X,17
461 PEEK(X)=94THENPOKE X,17
462 PEEK(X)=94THENPOKE X,17
463 PEEK(X)=94THENPOKE X,17
464 PEEK(X)=94THENPOKE X,17
465 PEEK(X)=94THENPOKE X,17
466 PEEK(X)=94THENPOKE X,17
467 PEEK(X)=94THENPOKE X,17
468 PEEK(X)=94THENPOKE X,17
469 PEEK(X)=94THENPOKE X,17
470 PEEK(X)=94THENPOKE X,17
471 PEEK(X)=94THENPOKE X,17
472 PEEK(X)=94THENPOKE X,17
473 PEEK(X)=94THENPOKE X,17
474 PEEK(X)=94THENPOKE X,17
475 PEEK(X)=94THENPOKE X,17
476 PEEK(X)=94THENPOKE X,17
477 PEEK(X)=94THENPOKE X,17
478 PEEK(X)=94THENPOKE X,17
479 PEEK(X)=94THENPOKE X,17
480 PEEK(X)=94THENPOKE X,17
481 PEEK(X)=94THENPOKE X,17
482 PEEK(X)=94THENPOKE X,17
483 PEEK(X)=94THENPOKE X,17
484 PEEK(X)=94THENPOKE X,17
485 PEEK(X)=94THENPOKE X,17
486 PEEK(X)=94THENPOKE X,17
487 PEEK(X)=94THENPOKE X,17
488 PEEK(X)=94THENPOKE X,17
489 PEEK(X)=94THENPOKE X,17
490 PEEK(X)=94THENPOKE X,17
491 PEEK(X)=94THENPOKE X,17
492 PEEK(X)=94THENPOKE X,17
493 PEEK(X)=94THENPOKE X,17
494 PEEK(X)=94THENPOKE X,17
495 PEEK(X)=94THENPOKE X,17
496 PEEK(X)=94THENPOKE X,17
497 PEEK(X)=94THENPOKE X,17
498 PEEK(X)=94THENPOKE X,17
499 PEEK(X)=94THENPOKE X,17
500 PEEK(X)=94THENPOKE X,17
501 PEEK(X)=94THENPOKE X,17
502 PEEK(X)=94THENPOKE X,17
503 PEEK(X)=94THENPOKE X,17
504 PEEK(X)=94THENPOKE X,17
505 PEEK(X)=94THENPOKE X,17
506 PEEK(X)=94THENPOKE X,17
507 PEEK(X)=94THENPOKE X,17
508 PEEK(X)=94THENPOKE X,17
509 PEEK(X)=94THENPOKE X,17
510 PEEK(X)=94THENPOKE X,17
511 PEEK(X)=94THENPOKE X,17
512 PEEK(X)=94THENPOKE X,17
513 PEEK(X)=94THENPOKE X,17
514 PEEK(X)=94THENPOKE X,17
515 PEEK(X)=94THENPOKE X,17
516 PEEK(X)=94THENPOKE X,17
517 PEEK(X)=94THENPOKE X,17
518 PEEK(X)=94THENPOKE X,17
519 PEEK(X)=94THENPOKE X,17
520 PEEK(X)=94THENPOKE X,17
521 PEEK(X)=94THENPOKE X,17
522 PEEK(X)=94THENPOKE X,17
523 PEEK(X)=94THENPOKE X,17
524 PEEK(X)=94THENPOKE X,17
525 PEEK(X)=94THENPOKE X,17
526 PEEK(X)=94THENPOKE X,17
527 PEEK(X)=94THENPOKE X,17
528 PEEK(X)=94THENPOKE X,17
529 PEEK(X)=94THENPOKE X,17
530 PEEK(X)=94THENPOKE X,17
531 PEEK(X)=94THENPOKE X,17
532 PEEK(X)=94THENPOKE X,17
533 PEEK(X)=94THENPOKE X,17
534 PEEK(X)=94THENPOKE X,17
535 PEEK(X)=94THENPOKE X,17
536 PEEK(X)=94THENPOKE X,17
537 PEEK(X)=94THENPOKE X,17
538 PEEK(X)=94THENPOKE X,17
539 PEEK(X)=94THENPOKE X,17
540 PEEK(X)=94THENPOKE X,17
541 PEEK(X)=94THENPOKE X,17
542 PEEK(X)=94THENPOKE X,17
543 PEEK(X)=94THENPOKE X,17
544 PEEK(X)=94THENPOKE X,17
545 PEEK(X)=94THENPOKE X,17
546 PEEK(X)=94THENPOKE X,17
547 PEEK(X)=94THENPOKE X,17
548 PEEK(X)=94THENPOKE X,17
549 PEEK(X)=94THENPOKE X,17
550 PEEK(X)=94THENPOKE X,17
551 PEEK(X)=94THENPOKE X,17
552 PEEK(X)=94THENPOKE X,17
553 PEEK(X)=94THENPOKE X,17
554 PEEK(X)=94THENPOKE X,17
555 PEEK(X)=94THENPOKE X,17
556 PEEK(X)=94THENPOKE X,17
557 PEEK(X)=94THENPOKE X,17
558 PEEK(X)=94THENPOKE X,17
559 PEEK(X)=94THENPOKE X,17
560 PEEK(X)=94THENPOKE X,17
561 PEEK(X)=94THENPOKE X,17
562 PEEK(X)=94THENPOKE X,17
563 PEEK(X)=94THENPOKE X,17
564 PEEK(X)=94THENPOKE X,17
565 PEEK(X)=94THENPOKE X,17
566 PEEK(X)=94THENPOKE X,17
567 PEEK(X)=94THENPOKE X,17
568 PEEK(X)=94THENPOKE X,17
569 PEEK(X)=94THENPOKE X,17
570 PEEK(X)=94THENPOKE X,17
571 PEEK(X)=94THENPOKE X,17
572 PEEK(X)=94THENPOKE X,17
573 PEEK(X)=94THENPOKE X,17
574 PEEK(X)=94THENPOKE X,17
575 PEEK(X)=94THENPOKE X,17
576 PEEK(X)=94THENPOKE X,17
577 PEEK(X)=94THENPOKE X,17
578 PEEK(X)=94THENPOKE X,17
579 PEEK(X)=94THENPOKE X,17
580 PEEK(X)=94
```

THE FORTE MUSIC PROGRAMMING LANGUAGE

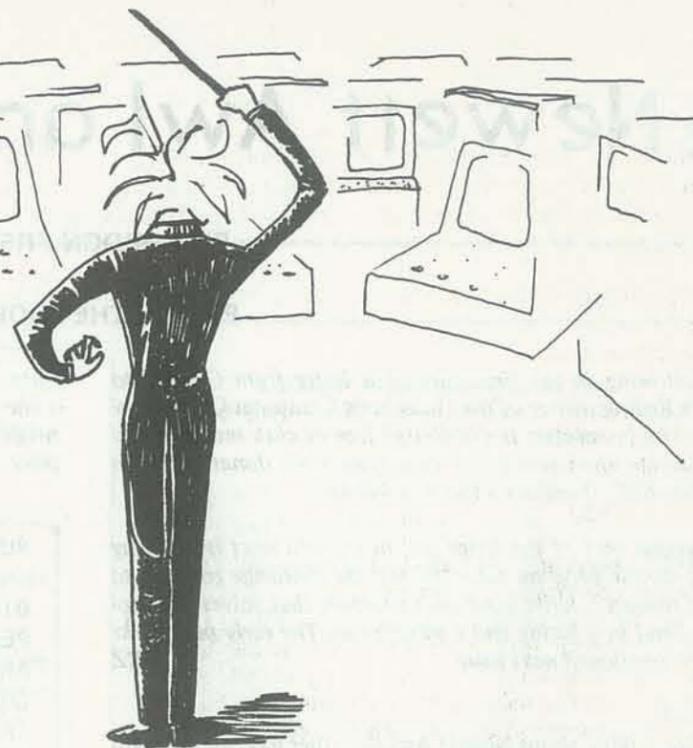
BY JIM DAY

In a note Jim sent with this excellent article, he remarks that he has used this language to code a wide range of music. His efforts have ranged from "Daisy" to the "Flight of the Bumblebee," despite the fact that he reads music with great difficulty.

"An impartial observer would say I don't read music at all," Jim adds. He also says, "No special hardware is needed, although a larger speaker sounds louder than the regular Apple beeper-peeper."

Near the part of the article that talks about sing-along lyrics there were some cryptic markings made by the Dragon. The Dragon mumblings included such stuff as "Lawrence Welk and Bubble Memories" and something about a "bouncing ball command." Oh, well! — RZ

Forte is a new music programming language developed by Gary J. Shannon. Currently implemented for the Apple II computer, Forte is available from SOFTAPE, 10756 Vanowen, North Hollywood, CA 91605 (213/985-5763). Designed for use by those familiar with standard musical notation but with little or no programming experience, Forte is very easy to learn. Its straightforward syntax and excellent editing and debugging capabilities make it a pleasure to use. A page of sheet music can be coded in Forte in a few minutes. In fact, it's easy to create Forte programs on-line, at the Apple keyboard, without writing code on paper.



Each note of a Forte program is represented by a letter (A through G), an octave number (1 through 8) and a duration number (1 through 64). A slash is used to separate the octave and duration numbers, and a pound sign (#) indicates a sharp. For example, the notation "C # 4/8" specifies a C-sharp eighth note in the fourth octave. A period (.) after the duration number causes the duration of a note to be extended by 50%, and an exclamation (!) indicates a triplet.

The only significant departure from the usual musical notation is Forte's convention whereby octaves begin with A and end with G#. This may take a few minutes, and a few mistakes, to get used to. Also a sharp applies only to the single note for which it is specified. The pseudo note R is used to specify a rest. Thus, R/4 indicates a quarter rest; R/8 indicates an eighth rest.

It isn't necessary to repeat all of the parameters for each note, if the octave and/or duration are the same for two or more notes in succession. For example, "A3/4 G3/4" could be coded as "A3/4 G" without confusing Forte.

Each line of code in a Forte program is identified by a decimal number, as in a BASIC program, and many notes can be written in the same line of code (up to 256 characters). Most Forte users find it convenient to write each measure of music as a single line of code, as this greatly simplifies debugging.



At the beginning of a Forte program one must specify the tempo and voice to be used. Tempo (T1 through T255) determines how fast the subsequent notes will be played. Normal tempo is about 175. Voice (V1 through V6) selects a predetermined envelope for the notes that follow. Tempo and/or voice may be respecified at any point in a program.

The P command is used to insert a pause of one or more seconds at any point in a program. For example, "P3" causes a pause of three seconds.

Comment lines are identified as such by a leading asterisk (*). Trailing comments in a line of procedural code are also allowed. For example, the line

```
100 A3/4 B C D *COMMENT
```

specifies four notes and includes a trailing comment. Comments have no effect on program execution.

If you want some text displayed on the screen by Forte while a program is running, you can simply include a quoted string at the point in the program where you want the text to appear. An H command can be used to clear the screen and home the cursor to the left-center of the display. For example, the line

```
100 H V2 T175 "DAISY"
```

would clear the screen, set voice two, set a tempo of 175, and print the DAISY, in that order. The display capability is useful for showing titles and sing-along lyrics.

Repetition of any part of a program is easily accomplished. The following line would play the notes A, B, C, and D a total of three times.

```
100 (3: A4/8 B C D)
```

The number preceding the colon in the example shown above specifies how many times the notes are to be played, and the parentheses indicate the sequence to be repeated.

The N command can be used to execute a line of code only during a given repetition of the program segment in which that line appears. Line 101 below would be executed only in the first performance of the parenthesized segment.

```
100 V1 T150 (2:
101 N1 A4/8 B C D
102 D C B A)
```

The J command is used to jump to any given line. For example, the following program would play the same four notes until stopped by the user.

```
100 V2 T100
101 B4/2 C# D D#
102 J101
```

Thus, a J command is equivalent to a GOTO in a BASIC program. Forte also provides the equivalents of GOSUB and RETURN. The U command has the effect of a GOSUB, and the X command has the effect of a RETURN. In the program shown below, line 132 calls a closed subroutine beginning at line 199. This example certainly isn't an optimum Forte program, since a linear structure would have served just as well, but it does illustrate how U and X commands can be used.

```
129 V3 T220 H "DAISY" *DISPLAY TITLE
```

```
132 U199 *CALL SUBROUTINE
```

```
135 Q *END MAIN PROGRAM
```

```
140 *
```

```
199 (3: *EXECUTE THREE TIMES
```

```
200 D4/4 B G3 D E/8 F3 G
```

```
201 E/4 G/8 D/2. A4/4. D B G3
```

```
202 E/8 F# G A4/4 B/8 A/4. R/4 B/8 C B A
```

```
203 D/4 B/8 A G3/2 A4/8 B/4 G3/8 E/4 G/8
```

```
204 E D/2 D/8 G/4 B4/8 A/4 D3/8 G3/4 B4/8
```

```
205 A/4 B/16 C D/8 B G3 A4/4 D3/8 G/2
```

```
206 R/2
```

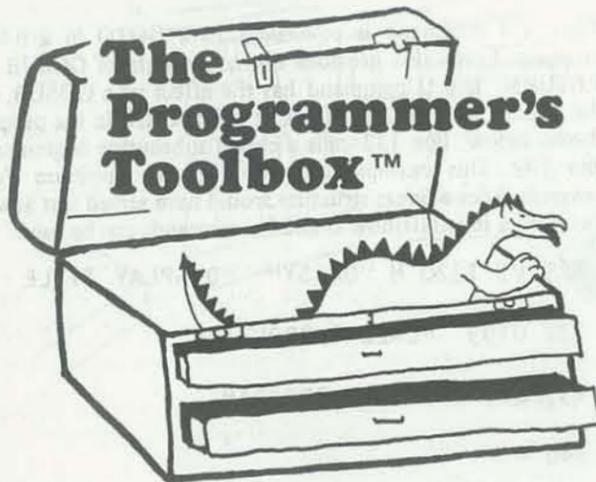
```
222 ) X *RETURN TO MAIN PROGRAM
```

Note that the Q command in line 135 above ends the main program. An S command has the same effect as a Q command, but is intended as a temporary breakpoint during debugging.

Direct commands are executed as soon as they are typed, and are not preceded by line numbers. The square-cornered brackets in the accompanying table of direct commands indicate optional parameters. Such brackets are not to be typed as part of any command.

Continued on page 31





BY EVERYBODY

In Vol. 1, No. 3 of PC, 1973, Marc Le Brun began a column that provided routines that could be used as part of a "toolbox" of computer skills. We revived that column beginning with the last issue of RC.

The Dragon (Bob Albrecht) saw the toolbox and immediately sent us the following contributions. He promises to send more. However, we also solicit material for the column from all of you. After all, you are everybody, too. — RZ

PT3: SCREEN OUTLINE

This TRS-80 birthday card includes a short subroutine to draw a line around the edge of the screen, using a single FOR-NEXT loop.

```
200 REM *** HAPPY BIRTHDAY 'CARD'
210 CLS
220 GOSUB 920
230 PRINT @ 348, "HAPPY!";
240 PRINT @ 539, "BIRTHDAY!";
250 PRINT @ 732, "MOTHER!";
260 GOTO 260

900 REM *** PROGRAMMER'S TOOLBOX #3: DRAW A BOX AROUND
910 REM *** THE OUTSIDE EDGE OF THE SCREEN
920 FOR KPT = 0 TO 127
930 SET (KPT, 0) : SET (KPT, 47)
940 IF KPT < 48 SET (0, KPT) : SET (127, KPT)
950 NEXT KPT
960 RETURN
```

BY THE DRAGON

PT4: PRINT@ SCROLLING

This TRS-80 program uses the PRINT@ statement and the bottom line scrolling feature of the TRS-80. Try it, then study lines 150, 160, 170 and 180 to figure out how and why it works as it does. Enjoy.

```
100 REM *** PROGRAMMER'S TOOLBOX (TM) NUMBER 4
110 REM *** RECREATIONAL COMPUTING, JUL/AUG 1979

120 CLS
130 X = RND (100)

140 PRINT @ 832, "I'M THINKING OF A NUMBER FROM 1 TO 100!"

150 PRINT @ 960, "GUESS MY NUMBER!"; : INPUT G
160 IF G < X PRINT @ 896 + 32, "TRY BIGGER!"; GOTO 150
170 IF G > X PRINT @ 896 + 32, "TRY SMALLER!"; GOTO 150

180 IF G = X PRINT @ 896 + 32, "THAT'S IT!!!"
190 FOR Z = 1 TO 700 : NEXT Z : GOTO 120
```

How does it work? Here are some hints.

- In line 150, print position 960 is the left edge of the bottom line of the screen.
- If we type something on the bottom line of the screen and press the ENTER key, everything on the screen will move up one place.
- In lines 160-180, print position 896 is the left edge of the line just above the bottom line of the screen. Print position 896 + 32 is about half way across that line.

BY THE DRAGON

PT5: REMOVING SPACES

Here is a little subroutine to remove all spaces from a string. You provide a string called APTS. Our subroutine removes all spaces and returns APTS to you, sans spaces.

In the subroutine, we have used line numbers 900 through 970. None of these line numbers are referred to within the subroutine, so change them to your heart's desire and write your GOSUBs accordingly.

The subroutine uses two other strings, BPTS and CPTS. CPTS is a string of length one (1). BPTS is used to build the "spaceless" string. In our line 970, APTS is set equal to BPTS, then BPTS is set equal to the null string of length zero. We do this to conserve string space in your program.

Now, does everyone out there know why we might want to remove all the spaces from a string?

```
200 REM *** ASK FOR STRING (APTS), REMOVE ALL CHARACTERS
210 REM *** WHICH ARE NOT UPPER CASE LETTERS, PRINT RESULT
220 CLS
230 INPUT "APTS = " : APTS
240 GOSUB 920
250 PRINT APTS : PRINT : GOTO 230
```

```
900 REM *** PROGRAMMER'S TOOLBOX # 6. REMOVE ALL CHARACTERS
910 REM *** WHICH ARE NOT UPPER CASE LETTERS FROM APTS
920 LPT = LEN (APTS) : IF LPT = 0 THEN RETURN
930 BPTS = ""
940 FOR KPT = 1 TO LPT
950 CPTS = MIDS (APTS, KPT, 1)
960 IF ASC (CPTS) > 64 AND ASC (CPTS) < 91 THEN BPTS = BPTS + CPTS
970 NEXT KPT
980 APTS = BPTS : BPTS = "" : RETURN
```

BY THE DRAGON

PT6: UPPER CASE ALPHABET SCAN

This subroutine removes all characters which are not upper case letters from the string APTS.

- For example, if you provide APTS = "ABC123I#S", the subroutine will return APTS = "ABC".

The subroutine uses two other strings, BPTS and CPTS. CPTS is a string of length one (1) consisting of individual characters from APTS. BPTS is used to build the string of upper case letters from APTS. In our line 980, APTS is set equal to BPTS, then BPTS is set equal to the null string of length zero. We do this to conserve string space in your program. □

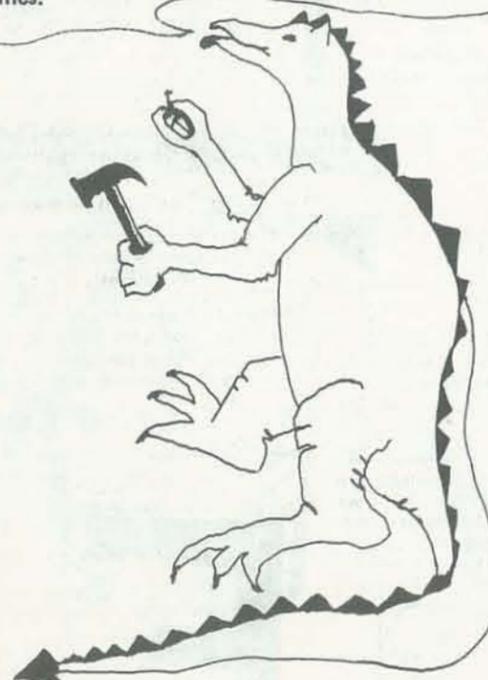
Anybody got a use for this subroutine?

```
200 REM *** ASK FOR STRING (APTS), REMOVE SPACES, PRINT RESULT
210 CLS
220 INPUT "APTS = " : APTS
230 GOSUB 910
240 PRINT APTS : PRINT : GOTO 220
```

```
900 REM *** PROGRAMMER'S TOOLBOX #5. REMOVE SPACES FROM APTS
910 LPT = LEN (APTS) : IF LPT = 0 THEN RETURN
920 BPTS = ""
930 FOR KPT = 1 TO LPT
940 CPTS = MIDS (APTS, KPT, 1)
950 IF CPTS < " " THEN BPTS = BPTS + CPTS
960 NEXT KPT
970 APTS = BPTS : BPTS = "" : RETURN
```

BY THE DRAGON

Extra Extra!
Attention Everybody!
The next RC toolbox will have stuff on the ATARI and TEXAS INSTRUMENTS Computers! Color, sound, graphics.



Continued from page 29

DIRECT FORTE COMMANDS

- CAS Play to tape rather than to speaker.
- CON Continue execution after stopping for an S command.
- DEL m[,n] Delete program lines m through n. A single line can also be deleted by typing its line number followed by a RETURN.
- FREE Display the number of bytes available for program expansion.
- LIST [m[,n]] List the current program in its entirety or lines m through n only.
- LOAD [name] Retrieve a program from tape, or from disk if a file name is specified.
- NEW Delete the current program, if any.
- NOTRACE Cancel the TRACE and/or TRACEN mode.
- PR# n Direct listing output to peripheral device n.
- RESTORE Cancel the most recent NEW command, unless program lines have been entered subsequent to that command.
- RUN [n] Start program execution, from the beginning or from line n.
- SAVE [name] Copy current program to tape, or to disk if a file name is specified.
- SPD n Set the speed of listing a program, where n may be from 1 (very slow) to 255.
- SPK Return audio output to the speaker, if a CAS command is in effect.
- STEP [n] Play one note at a time, from the beginning or from line n, as the space bar or any alphanumeric key is pressed. The RETURN key cancels the STEP mode.
- TRACE Display the number of each line as it is executed.
- TRACEN Display each note as it is played. □

FORTMAN

BY LEE SCHNEIDER & TODD VOROS

Volume III
Episode 8

In our last episode, Our Hero had just RETURN'ed from a successful mission to the FIFO fortress, accompanied by Linea of the Resistance and the Lockout Monster. A well-timed BREAK character had been executed in order to free those resistance elements captured in a recent

As they approach the threshold of those gates, their presence is detected and a response signal is generated...

You there! Execute an immediate HALT and state your ID and Job Number!

I am specified as Linea... and I and my resistance decades have come to...

Resistance! Ach, be off with ye... we are well enough at ohm here, and have no need for the likes of more resistance! There be no time for us to spend on transients!

Er... not even if we come to return a great treasure to your Clan?

All right... I'll spare ye a few microcycles. Now, what's all this about a treasure? Fetch it on out and let's decode its instruction value!

But... it's already here, sir. You mean you don't recognize your very own Lockout Monster?

gleep?

eep!

Obviously the little creature is overjoyed at being reunited with its old master program. Yet the elder McIntel still seems confused by a most significant bit...

And the master of the Clan has more to tell them...

Ye know... we of the Clan have not remained at neutral potential all this time by choice... it was because with our Monster in the hands of the Glitchmaster, we were locked out of our own lands! But now...

I only hope it isn't too late! And if there be anything we could do for you to RETURN the favor of your CALL...

For a very simple function, sir... to restore the missing memory to our comrade here!

Hmmm... I suppose... He must be a high-priority block of code to warrant such service!

Well... er... yes, I see... but, what happened to the poor beastie? My memory seems to hold an image of a somewhat... er, larger... module!

Don't worry, sir. Despite his recently altered DIMENSIONS he's as good as ever... a condition I'm sure the guards at FIFO fortress will verify!

But there is... something that would not only help us but our land as well! Tell me; is your In-Circuit Emulator running?

sigh

MISSING MEMORY

Aye... that it is... but what for?

battle... a battle in which, if not for the incredible powers of F-Man, all would have been terminated for sure!

But there is no time for celebration. In faraway Capital City, the main force of the Resistance under General Wirewound is preparing for the final assault on the stronghold of the Glitchmaster... and if Linea cannot re-position her resistance force in time to parallel the General's attack, this last valiant attempt to restore noise-free order to Microprocessorland is doomed to be a non-repeatable hardware failure!

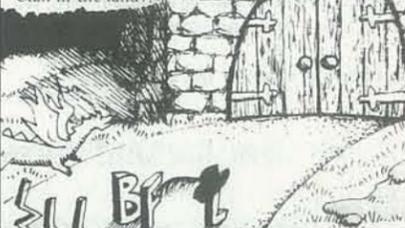
Yet with all his computational power, Fortman cannot help them. When part of his memory was inadvertently erased, he forgot who he was and how to use most of those powers! And not even those resistance elements around him know the true file name of his code.

But wait! Among those segments freed when FIFO was emptied there pops up none other than F-Man's partner in computer crime-fighting, Billy Basic. Billy quickly reveals to an amazed resistance who this stranger really is!

Linea immediately realizes that F-Man could provide the solution to their complex problems... if only his memory could be re-loaded! And there is only one way that could be done...

So across the great data fields march the forces of the Resistance, towards the Great Voltage divide... and Castle McIntel!

Crossing the precariously balanced bridge formed by the resistance over the Voltage Divide, our group at last approaches the very gates of the Castle... the ancestral stronghold of Clan McIntel, the oldest and most populous Clan in the land!



There is a slight delay of perhaps a microsecond or two... then the great gates switch open and out propagates none other than Angus McIntel himself, leader of the Clan!

Recreational COMPUTING

SUBSCRIPTION ORDER

Please send me a one-year subscription to **Recreational Computing** magazine, formerly *People's Computers* (published bi-monthly) for \$10.

NAME _____

ADDRESS _____

CITY/STATE _____ ZIP _____

Check enclosed Bill me (U.S. only)

Signature _____

Renewal (please attach mailing label)

Charge my card: Visa/Bankamericard Mastercharge

Card No. _____ Exp. Date _____

Signature _____

International Rates: Canada *First Class* \$17 One Year

Rest of World *Airmail* \$23 One Year

World *Surface Mail* - I will risk the lengthy and unreliable delivery of surface mail (signed)

\$14 One Year

Payment must be in \$US drawn on a US Bank.

C3

Exp. Date: 12/31/79

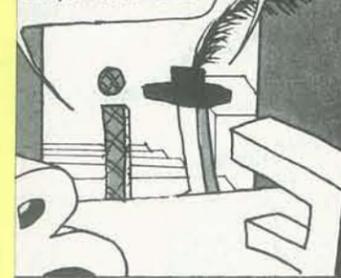
A4

Exp. Date: 12/31/79

send a serial message serial, instructing him to battle?

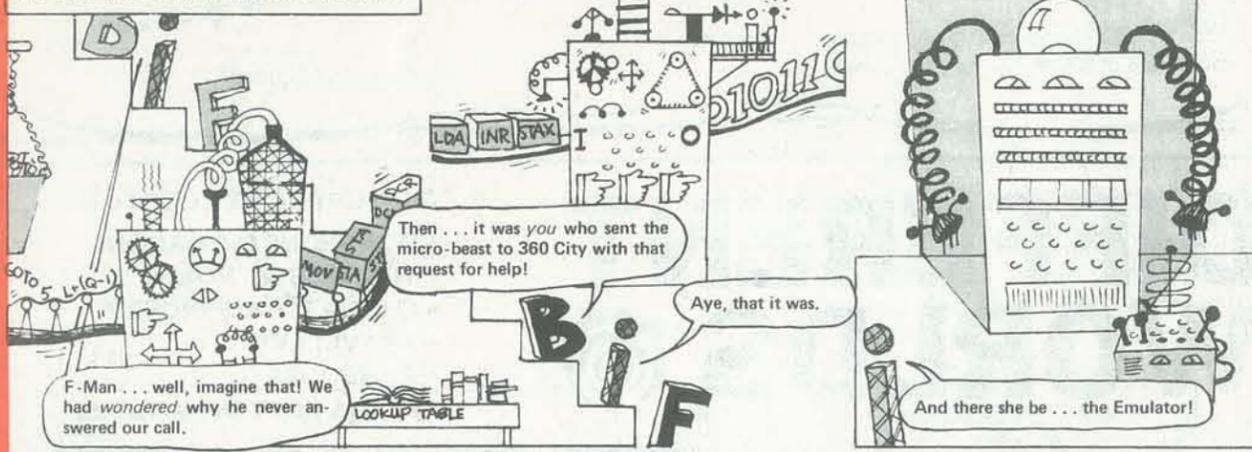
I already thought of that... and I dare not! The Glitchmaster could all too easily alter such a message, and make it unreadable... or worse! No... the only way we can support our fellow resistance units is with positive-supply action!

en... if ye wish to take everyone down to the



Finally they reach the lowest floor of hardware level. And here, amongst the layer of digital dust, there stands a strange ent device.

Separators process long, complex statements and sift the results into small mnemonic blocks...



Then... it was you who sent the micro-beast to 360 City with that request for help!

Aye, that it was.

F-Man... well, imagine that! We had wondered why he never answered our call.

And there she be... the Emulator!

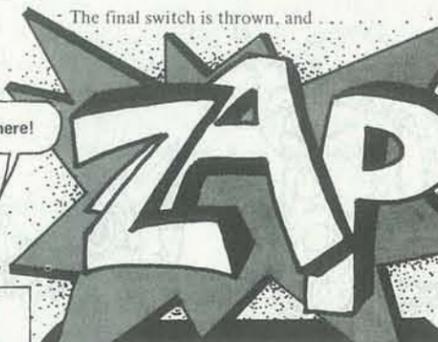
Angus McIntel wastes no cycles, as he swiftly aligns Billy under the data ports of the emulator.

All right now... are ye ready?

The final switch is thrown, and...

Er... I guess so.

Ready here!



Will Linea's plan work? Will F-Man be restored by the operation, and at last regain his lost memory data? Will Billy survive the transfer - or will he suffer from Basic disorganization? Will the Resistance ever succeed in pulling down the Glitchmaster's evil and noisy regime?

There we go... have to watch out for the right pinout, you know!

The interconnections are complete, the power is switched on, and final preparations are made for the memory-to-memory data transfer.

For the answer to these and other non-structured questions, tune in for the next episode same statement, same line number.

FORTRAN MAN

BY LEE SCHNITZER & TODD VOIGT

Volume III
Episode 8

In our last episode, Our Hero had just returned from a successful mission to the FIFO accompanied by Linea of the Resistance, the Lockout Monster. A well-timed character had been executed in order those resistance elements captured in

As they approach the threshold of the fort, their presence is detected and a response is generated...

You there! Execute an immediate HALT and state your ID and Job Number!

I am specified as Linea... and I and my resistance decades have come to...

Resistance! Ach, be off with ye... we are well enough at ohm here, and have no need for the likes of more resistance! There be no time for us to spend on transients!

Er... not even if we come to return a great treasure to your Clan?

gleep?

But... it's already here, sir. You mean you don't recognize your very own Lockout Monster?

All right... I'll spare ye a few microcycles. Now, what's all this about a treasure? Fetch it on out and let's decode its instruction value!

esp!

Obviously the little creature is overjoyed at being reunited with its old master program. Yet the elder McIntel still seems confused by a most significant bit...

slurp slurp slurp

Well... er... yes, I see... but, what happened to the poor beastie? My memory seems to hold an image of a somewhat... er, larger... module!

Don't worry, sir. Despite his recently altered DIMENSIONS he's as good as ever... a condition I'm sure the guards at FIFO fortress will verify!

And the master of the Clan has more to tell them...

Ye know... we of the Clan have not remained at neutral potential all this time by choice... it was because with our Monster in the hands of the Glitchmaster, we were locked out of our own lands! But now...

I only hope it isn't too late! And if there be anything we could do for you to RETURN the favor of your CALL...

But there is... something that would not only help us but our land as well! Tell me; is your In-Circuit Emulator running?

Missing Memory

Aye... that it is... but what for?

For a very simple function, sir... to restore the missing memory to our comrade here!

Hmmm... I suppose... He must be a high-priority block of code to warrant such service!

sigh

DR. DOBB'S JOURNAL of COMPUTER Calisthenics & Orthodontia

NO POSTAGE NECESSARY IF MAILED IN THE UNITED STATES

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO. 756 MENLO PARK, CA

POSTAGE WILL BE PAID BY ADDRESSEE

DR. DOBB'S JOURNAL
P.O. BOX E
1263 EL CAMINO REAL
MENLO PARK, CA 94025

He could win the battle for us. This file, sir, is none other than the one and only Fortran Man!

Carefully Angus considers the option...

Couldn't ye send a serial message to the General, instructing him to hold off the battle?

I already thought of that... and I dare not! The Glitchmaster could all too easily alter such a message, and make it unreadable... or worse! No... the only way we can support our fellow resistance units is with positive-supply action!

All right then... if ye wish to take the chance; everyone down to the lab!

What? F-Man, ye say! But then... how do ye propose to do such a restore? We have no Fortran records here in Micro Land.

But we do have my friend Billy Basic... perhaps the greatest authority on F-Man anywhere... and his knowledge will serve as source file to replace F-Man's memories!

Yes... it might work... but are ye sure ye want to risk that? I mean putting a high-level language through an emulator could be dangerous!

We know... but it's our only chance to save microprocessorland! Even as we interchange data here, General Wirewound is starting the attack on Capital City, and without our help...!!!

Swiftly they branch through the gates and into the castle, down the staircase generator into the masses of McIntel Hardware. Down they go... past the compiler level, where huge Syntax Separators process long, complex statements and sift the results into small mnemonic blocks...

Down past assembly level, where the blocks are re-formed into the most basic machine-level codes, bit by bit...

And finally they reach the lowest floor of all... hardware level. And here, amongst the fine layer of digital dust, there stands a strange and silent device.

Then... it was you who sent the micro-beast to 360 City with that request for help!

Aye, that it was.

F-Man... well, imagine that! We had wondered why he never answered our call.

LOOKUP TABLE

And there she be... the Emulator!

Angus McIntel wastes no cycles, as he swiftly aligns Billy under the data ports of the emulator.

All right now... are ye ready?

Er... I guess so.

Ready here!

ZAP

The final switch is thrown, and...

Will Linea's plan work? Will F-Man be restored by the operation, and at last regain his lost memory data? Will Billy survive the transfer - or will he suffer from Basic disorganization? Will the Resistance ever succeed in pulling down the Glitchmaster's evil and noisy regime?

Good luck, Billy... the Land of the Little People is depending on you!

The interconnections are complete, the power is switched on, and final preparations are made for the memory-to-memory data transfer.

There we go... have to watch out for the right pinout, you know!

For the answer to these and other non-structured questions, tune in for the next episode same statement, same line number.

Paedia
The Computing Sisterhood
School without a place!

futureplay™

Holistic Computing
Don't Bother Me . . . I'm Learning
Computertown, U.S.A.!

Project Critical Mass
Have you heard Herb Moore's music?

Three-year-old kids at the keyboard
Computers & Libraries
Computers & Pizza
Computers & Champagne
Computers and . . .

What is Art Canfil (Taipan) up to now?
Down Home Computer Book
New Ways of Learning

PETAL
ADA
(neither dentists nor politicians)
Since schools won't change . . . abandon them!




Stuff on ATARI & TI Home Computers
West Coast Conference of NPCO
Can GANDALF be the language of FuturePlay?

"Don't try to change existing institutions — invent new ones . . ."
Paraphrase of a Bob Theobald statement
Games Books Galore

DIFFERENT WORLDS

FEATURING ARTICLES ON:

- * DUNGEONS & DRAGONS
- * TUNNELS & TROLLS
- * CHIVALRY & SORCERY
- * TRAVELLER
- * RUNEQUEST!
- * AND A HOST OF OTHERS!

An authoritative journal that no dedicated gamemaster should do without! Collect *Different Worlds* . . . treasure each issue! Appearing six times a year, it features informative and timely articles. *Different Worlds* is The Magazine of Game Role-Playing for Game Role-Playing Fans! Subscribe now!

 A NEW MAGAZINE OF GAME ROLE-PLAYING 

SUBSCRIPTION RATES

U.S. Subscriptions \$10.50/1 year (6 issues)
Canada and Mexico \$14.50
First Class
Rest of World Surface \$14.50
Rest of World Air Mail \$21
Payment must be in US funds.

DIFFERENT WORLDS • PO BOX 6302 • ALBANY, CA 94706



ZORK: A COMPUTERIZED FANTASY SIMULATION GAME

BY P. DAVID LEBLING
MARC S. BLANK
TIMOTHY A. ANDERSON

MIT LABORATORY FOR COMPUTER SCIENCE

Small versions of large-scale games are beginning to appear on personal computers. For example, we just received a TRS-80 cassette of Adventure from Scott Adams, P.O. Box 3435, Longwood, FL 32750.

By reprinting this article on Zork, we hope to interest a few people in perhaps producing a "Tiny Zork" for our home computers. With Adventure in hand, can Zork be far behind?

The authors of this article have a variety of interests and skills in addition to their fascination with Zork. P. David Lebling is a staff member of MIT Laboratory for Computer Science; Marc S. Blank is a medical student at the Albert Einstein College of Medicine; Timothy A. Anderson is a member of the research staff at Computer Corporation of America, Cambridge, MA.

Happy Zorking!!

— RZ



Welcome to Zork.
This version created December 8.
West of House
You are in an open field west of a big white house with a boarded front door.
There is a small mailbox here.
> GO NORTH
North of House
You are facing the north side of a white house. There is no door here, and all the windows are barred.
> EAST
Behind House
You are behind the white house. In one corner of the house there is a small window which is slightly ajar.
> OPEN THE WINDOW
With great effort, you open the window far enough to allow entry.
> GO IN
Kitchen
You are in the kitchen of the white house. A table seems to have been used recently for the preparation of food. A passage leads to the west, and a dark staircase can be seen leading upward. To the east is a small window which is open.
On the table is an elongated brown sack, smelling of hot peppers.
A bottle is sitting on the table.
The glass bottle contains:
A quantity of water

Beyond this nondescript kitchen, above and below the surface of the earth, lie scores of rooms, some containing traps, some containing puzzles. Hundreds of objects are scattered throughout this maze, some valuable treasures, some magical tools. The little white house in the forest clearing is the entrance to Zork, a game developed by the authors. Zork is one example of a new type of game: the Computerized Fantasy Simulation game.

In this type of game, the player interacts conversationally with an omniscient "Master of the Dungeon," who rules on each proposed action and relates the consequences. If the player says "Go north," he may move north, or the dungeon master may say "There is no way to go in that direction." If the player says "Open the window," the dungeon master may respond "The window is locked." The results depend on the design of the game, its architecture and furnishings, so to speak: in one game picking a sword might be fatal; in another it might confer magical powers. The design and implementation of such games is as much an exercise in creative writing as in programming.

The interest in playing Zork (or any other CFS game) is two-fold. First, the object of the game is usually to collect treasure, and this may be done only by solving problems; in the above the player would garner 10 points by being clever enough to open the window and enter the house. (Zork itself has more than two dozen distinct problems to solve, some presented in several stages.) Second, a great deal of the enjoyment of such games is derived by probing their responses in a sort of informal Turing test: "I wonder what it will say if I do this?" The players (and designers) delight in clever (or unexpected) responses to otherwise useless actions.

Overview: Simulating the Universe

The heart of any CFS game is its ability to mimic omniscience. By this we mean that the game should simulate the real world sufficiently well so that the player is able to spend most of his time solving the problems rather than solving the program. If, for example, the vocabulary is too small, the player must always wonder if his problem is only that he hasn't yet found the right word to use. Similarly, it is annoying for a possible solution to a problem to be ignored by the game. In other words, the program must simulate the real world.

Obviously, no small computer program can encompass the entire universe. What it can do, however, is simulate enough of the universe to appear to be more intelligent than it really is. This is a successful strategy only because CFS games are goal-directed. As a consequence, most players try to do only a small subset of the things they might choose to do with an object if they really had one in their possession.

Zork "simulates the universe" in an environment containing 191 different "rooms" (places to be) and 211 "objects." The vocabulary includes 908 words, of which 71 are distinct "actions" it handles. By contrast, a person's conversational vocabulary is a factor of two (or more) larger. How, then, does a limited program make a creditable showing in the conversational interaction that characterizes Zork?

The technique Zork uses for simulating the universe is that of universal methods modified for particular situations. For example, when a player tries to take some object, he expects to end up carrying it. There are, as in the real world, exceptions: some objects are "nailed down," and one's carrying capacity is limited. These restrictions are included in the general TAKE function. However, the designer might want a special action in addition to, or instead of, the general TAKE: a knife might give off a blinding light when taken; an attempt to take anything in a temple might be fatal. These exceptions would not appear in the general TAKE function, but in functions associated with the knife and the temple.

The details of this method of exceptions will be taken up later. The effect of it is that "the expected thing" usually happens when the player tries to (for example) take something. If the object he is trying to take is not special, and the situation is not special, then "it works," and he gets the object. In Zork, there are quite a few of these basic verbs. They include "take," "drop," "throw," "attack," "burn," "break," and others. These basic verbs are set up to do reasonable things to every object the player will encounter in the game. In many cases, objects have properties indicating that a certain verb is meaningful when applied to them (weapons have a "weapon" property, for example, that is checked by the verb "attack"). Applying a verb to an object lacking the necessary property often results in a sarcastic retort. ("Attacking a troll with a newspaper is foolhardy.") But the point is that it does something meaningful, something the player might have expected.

Another way in which the game tries to be real is by the judicious use of assumptions in the dungeon master's command parser. Suppose the player says "Attack." Assuming that he has a weapon and there is an enemy to attack, this should work, and it does. Assumptions are implemented by the existence of verb frames (stereotypes) and memory in the parser.

In the example, the parser picks up the verb frames for the verb "attack." They indicate that "Attack 'villain' with 'weapon'" is the expected form of the phrase. Now, "villain" means another denizen of the dungeon, so the parser looks for one that is presently accessible, a "villain" in the same room as the player. Similarly, the player must have a "weapon" in his possession. Assuming only one "villain" is in the room and the player has only one "weapon," they are placed in the empty slots of the frame and the fight is on.

Suppose that there is only one villain available, the troll, but the player has two weapons: a knife and sword. In that case, the dungeon master can't decide for him which to use, so it gives up, saying "Attack troll with what?" However, it remembers the last input, as augmented by the defaults ("Attack troll"). Thus, if the user replies "With sword," or even "Sword," it is merged with the leftover input and again the fight is on. This memory can last for several turns: for example, "Attack"; "Attack troll with what?"; "With knife"; "Which knife?"; "Rusty knife"; and so on.

Data Structure and Control Structure

The underlying structure of Zork consists of the data base (known as "the dungeon") and the control structure. The data base is a richly interconnected pointer structure joining instances of four major data types: "rooms," locations in the dungeon; "objects," things that may be referenced; "actions," verbs and their frame structures; and "actors," agents of action. Each instance of these data types may contain a function which is the specializing element of that instance. The control structure of Zork is, at one level, a specification of which of these functions is allowed to process an input sentence and in what order.

In the simplest sense, Zork runs in a loop in which three different operations are performed: command input and parsing, command execution, and background processing. (Figure 1 is a flowchart of the Zork program.)

The command input phase of the loop is relatively straightforward. It is intended to let the user type in his command, edit it if he needs to, and terminate it with a carriage return.

The purpose of the Zork parser is to reduce the user's input specification (command) to a small structure containing an "action" and up to two "objects" where necessary.

The parser begins by verifying that all the words in the input sentence are in its vocabulary. Then, it must determine which action and objects, if any, were specified. For an object to be referenced in a sentence, it must be "available"—that is, it must be in the player's possession, in the room the player is in, or within an object that is itself available. Objects not meeting these criteria may still be referenced if they are "global objects," which are of two types: those that are always available (such as parts of the player's body), and those that are available in a restricted set of rooms (such as a forest or a house). Adjectives supplied within the sentence are used to distinguish objects of the same generic type (such as knives and buttons) but are otherwise ignored. If an object remains ambiguous, the parser asks which of the ambiguous objects was meant (for example, "Which button should I push?").

Next is syntax checking, whereby the correct "action" is used for any verb. Syntax checking makes use of any supplied prepositions, differentiating between, for example, "look at" and "look under," which imply different actions. Finally, having determined the appropriate syntax for a given sentence, the parser ensures that all required objects for a given action were specified. The parser may, for example, decide that the correct syntax for the sentence "Give X" is "Give X to Y." An attempt is then made to supply an appropriate "Y" to complete the sentence. This is made possible by the definitions of the actions themselves, which include the attributes of the objects to be acted upon. In the present example, the action for "Give" defines the indirect object ("Y") to be another denizen of the dungeon; the parser attempts to comply by seeing if one is available. If so, the indirect object is found, and the parse is successful. If not, the player is asked to supply the indirect object himself. ("Give X to whom?") Once this phase is completed, the parse is finished and the parser's output is returned.

The adjectives and prepositions that were in the user's input are used only to determine the "action" and the "objects," and are not part of the parser's output. In addition, all articles are ignored, though users may add them to increase the clarity (to themselves) of what they input. For example, an input of "Knock down the thief with the rusty knife" reduces to something like

```
[<action STRIKE> <object THIEF>
<object RUSTY-KNIFE>]
```

If, however, the input were "Knock on the thief," the parser would reduce that to

```
[<action RAP> <object THIEF>]
```

recognizing that the "action" to be performed depends, for the word "knock," on the syntax of the input sentence: "knock down" turns into "strike," while "knock on" turns into "rap."

Once parsing has been completed, processing of the command is started. The functional element (if any) of each of the objects in the parsed input may be invoked. Additionally, some objects not specifically referenced, but which define the situation, are part of the processing sequence. The order in which these functions are invoked is determined by a strategy of allowing the exceptions an opportunity to handle the input before performing the action associated with the most general case. The processing order is as follows:

- The actor performing the command, if any. This allows, for example, a robot with a limited range of actions.
- The vehicle the actor is in, if any. This allows the vehicle to restrict movement. For example, inside a freely drifting balloon the normal movement commands (such as "Run north") might be meaningless or even fatal.
- The verb, or "action."
 - (a) The indirect object of the sentence, if any.
 - (b) The direct object of the sentence, if any.
- The vehicle again, if any. The vehicle is called a second time to enable it to act based on changes in the state resulting from the action.
- The room the player is in.

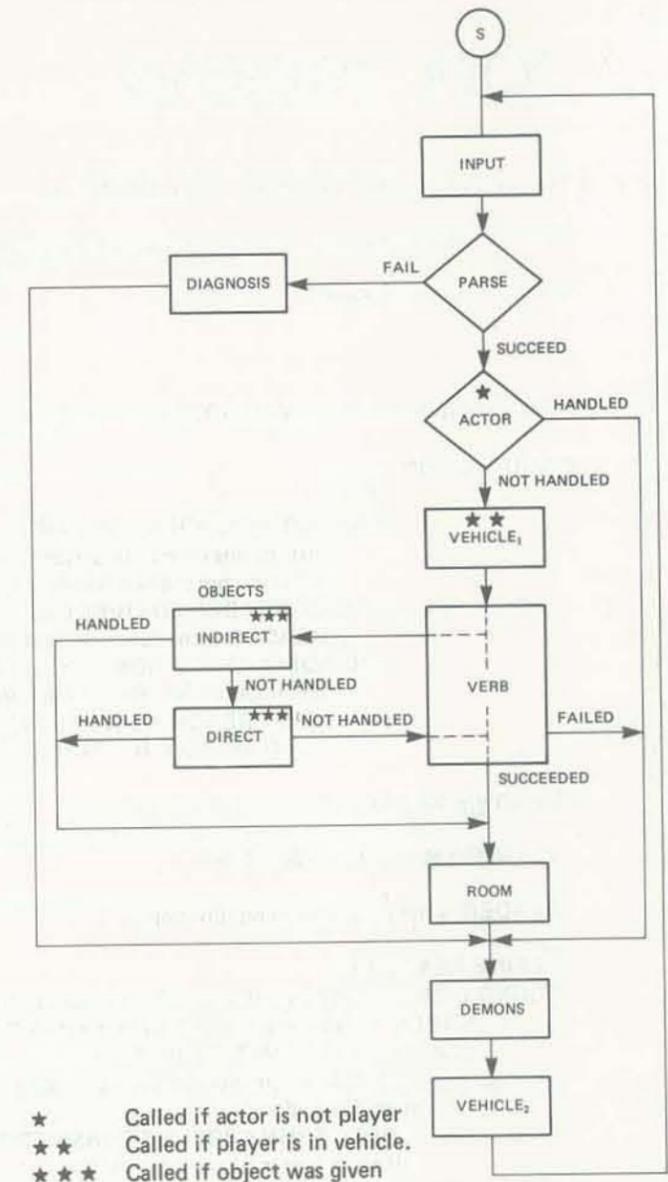


Figure 1. Zork flowchart.

Each of these functions is invoked in turn and given the option of handling the command. If it decides to handle the command, processing terminates at that point, and the remaining functions are not invoked. Otherwise, the sequence continues. Note that a function may do something (such as print a message) without completely handling the input. The invocation of an object's function is under the control of the verb; it may, after suitable checks, determine that the player's request is not reasonable. ("Your load is too heavy. You will have to leave something behind.") This limits flexibility slightly, but it has the advantage that it localizes the tests for a reasonable state.

ZORK INTERNALS

The following are some example Zork internals.

Comments (as in the MDL language) are strings following a semicolon.

Thus ;"I am a comment."

;"The definition of the 'verb' READ:"

```
<ADD-ACTION "READ"
  "Read"
  [(,READBIT REACH AOBJS ROBJs TRY)
    ;"restrictions on characteristics and location of objects for defaulting—filling in an unadorned
    'READ' command. The object must be readable and accessible."
  ["READ"READER] DRIVER]
  ;"READER is the function, and the form 'READ object' is preferred (the 'driver')"
  [(,READBIT REACH AOBJS ROBJs TRY) "WITH" OBJ ["READ"READER]]
  ;"specification for 'READ obj1 WITH obj2' "
  [(,READBIT REACH AOBJS ROBJs TRY) "THROU"OBJ["READ"READER]]
  ;"specification for 'READ obj1 THROUGH obj2' " >
```

;"Synonyms for READ:"

```
<VSYNONYM "READ" "SKIM">
```

;"READER is the general reading function:"

```
<DEFINE READER ( )
  <COND (<NOT <LIT? ,HERE>> ;"There must be light to read."
    <TELL "It is impossible to read in the dark.">)
    (<AND <NOT <EMPTY? <PRSI>>>
      ;"<PRSI> is the indirect object. If there is one, the player is trying to use something as a
      magnifying glass."
      <NOT <TRNN <PRSI>, TRANSBIT>>
      ;"If so, it better be transparent!">
      <TELL "How does one look through a " 1 <ODESC2 <PRSI>> "?">
      ;"It failed the test, so print sarcasm")
      (<NOT <TRNN <PRSO>, READBIT>>
        ;"The direct object should be readable."
        <TELL "How can I read " 1 <ODESC2 <PRSO>>"?">
        ;"It's not.")
      (<OBJECT-ACTION>
        ;"Now the object read gets a chance.")
      (ELSE ;"It didn't handle it, so print text."
        <TELL <OREAD <PRSO>>, LONG-TELL 1)>>>
```



;"An object: A stack of Zorkmid bills (Zorkmids are the currency of Zork)"

```
<OBJECT ["BILLS" "STACK" "PILE"]
  ;"The object's name and synonyms."
  ["NEAT" "200" "ZORKM"]
  ;"Adjectives which refer to the object."
  "stack of zorkmid bills"
  <+, OVISION, READBIT, TAKEBIT, BURNBIT>
  ;"Properties of the object: it's visible, readable, takeable, flammable"
  BILLS-OBJECT
  ( ) ;"The contents of the object (always empty for this object)."
  [ODESC1
    "200 neatly stacked zorkmid bills are here."
    ;"The long description."
  ODESCO
    "On the floor sit 200 neatly stacked zorkmid bills."
    ;"The initial long description (when first seen by the player)."

```

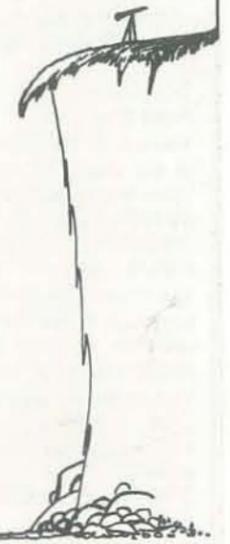
;"The object function for the Zorkmid bills. It is there mostly to make a few sarcastic comments."

```
<DEFINE BILLS-OBJECT ( )
```

```
  <SETG BANK-SOLVEI-FLAG T>
  <COND (<VERB? "BURN">
    <TELL "Nothing like having money to burn!">
    <> ;"Prints sarcasm but doesn't handle the command (accomplished by returning the false
    object <>). This allows BURN to also deal with it."
    (<VERB? "EAT">
      <TELL "Talk about eating rich foods!">
      ;"Doesn't allow EAT to run (by returning a non-false.)">>>
```

;"A room: the vault in which the Zorkmid bills are found"

```
<ROOM "BKVAU" ;"The internal name of the room."
  "This is the Vault of the Bank of Zork, in which there are no doors."
  "Vault"
  ,NULEXIT ;"There are no exits from this room."
  (<GET-OBJ "BILLS">)
  ;"The bills are initially here."
  <>
  <+, RSACREDBIT, RLANDBIT>
  ;"The room may not be entered by the thief, and is a land room."
  [RGLOBAL <+, WALL-ESWBIT, WALL-NBIT>]>
  ;"The walls of the room are/may be referenced."
```



Presumably, one of the functions will handle the command and print an appropriate response. Should that not happen, the response "Nothing happens" is printed by default. However, care has been taken to ensure that most input commands produce some reasonable response. Indeed, much of the enjoyment of the game is in being allowed to try ridiculous things, and the surprise of having the game understand them.

The functions described so far are invoked in direct response to what the user typed. Background processes, or "demons," are invoked after each input, regardless of its nature. They allow the program to do things independently of the player's actions.

Currently, there are four demons. The first is the "fighting" demon. The residents of the dungeon are frequently hostile; this demon allows them to assault the player unprovoked, and to keep fighting him even if he ignores them.

Next is the driving process behind the "thief," described as a "seedy looking gentleman carrying a large bag." The thief's purpose is to make life difficult for the player by absconding with treasures or other randomly selected objects. In many ways he acts like another, rather hostile and powerful, player in the dungeon.

The third demon is used to warn the player of the presence of hostile forces by causing his sword (if he has it) to glow if there are enemies nearby. It looks at the player's vicinity and prints an appropriate message if the "state of alert" changes; since the thief moves on his own, it is not sufficient to look for hostiles when the player moves.

Last is the "clock" demon. It is the mechanism by which the concept of future time is introduced into the game; arbitrary events can be scheduled for arbitrary future times. For example, the lamp can burn out after being on for some number of moves, and wounds inflicted in a fight will eventually heal. Out of consideration for poor typists, the clock does not tick after unparsed input.

The History of Zork

The existence of Zork is a direct consequence of the existence of two excellent games: Dungeons and Dragons, a fantasy simulation game (not computer based) invented by Dave Arneson and Gary Gygax, and Adventure, a computerized fantasy simulation game originally written by Wil Crowther and later extensively expanded by Don Woods.

Adventure itself was inspired by D&D (as it is familiarly known), in particular a D&D variation then being played out at Bolt, Beranek, and Newman, a Cambridge, Massachusetts, computer firm. It eventually was released to the public, and it became one of the most popular computer games in recent memory.

One laboratory that acquired a copy of Adventure was MIT's Laboratory for Computer Science, with which the designers of Zork (the authors and Bruce K. Daniels) were all then affiliated. In the process of "solving" Adventure, however, the game's deficiencies and the competitive spirit that often animates computer researchers kindled the desire of the authors to write a successor game.

Our natural choice of language was MDL, which is one of the languages in use at LCS. MDL recommended itself for other reasons, however. It is a descendent of LISP and is functionally extensible. It also permits user-defined data types, which is important in a game of "rooms," "objects," "verbs," and "actors." Finally, MDL makes it easy to imbed implicit functional invocations in data structures to tailor the game as described above. The initial version of the game was designed and implemented in about two weeks.

The first version of Zork appeared in June 1977. Interestingly enough, it was never "announced" or "installed" for use, and the name was chosen because it was a widely used nonsense word, like "foobar."

The original version of the game was much smaller, both geographically and in its capabilities. Various new sections have prompted corresponding expansions in the amount of the universe simulated. For example, the need to navigate a newly added river prompted the invention of vehicles (specifically, a boat). Similarly, the addition of a robot prompted the invention of other actors than the player himself: beings that could affect their surroundings, and so on. Fighting was added to provide a little more randomness in a fairly deterministic game.

The Future of Computer Fantasy Simulation Games

Zork itself has nearly reached the practical limit of size imposed by MDL and the PDP-10's address space. Thus the game is unlikely to expand (much?) further. However, the substrate of the game (the data types, parser, and basic verbs) is sufficiently independent that it would not be too difficult to use it as the basis for a CFS language.

There are several ways in which future computerized fantasy simulation games could evolve. The most obvious is just to write new puzzles in the same substrate as the old games. Some of the additions to Zork were exactly this, in that they required little or no expansion of the simulation universe. A sufficiently imaginative person or persons could probably do this indefinitely.

Another similar direction would be to change the milieu of the game. Zork, Adventure, and Haunt (the CFS games known to the authors) all flow back to D&D and the literary tradition of fantasy exemplified by J.R.R. Tolkien, Robert E. Howard, and Fritz Leiber. There are, however, other milieus; science fiction is one that comes to mind quickly, but there are undoubtedly others.

A slightly different approach to the future would be to expand the simulation universe portrayed in the game. For example, in Zork the concept of "wearing something" is absent: with it there could be magic rings, helmets, boots, etc. Additionally, the player's body itself might be added. For example, a player could be wounded in his sword arm, reducing his fighting effectiveness, or in his leg, reducing his ability to travel.

The preceding are essentially trivial expansions to the game. A more interesting one might be the introduction of magic spells. To give some idea of the kinds of problems new concepts introduce to the game, consider this brief summary of

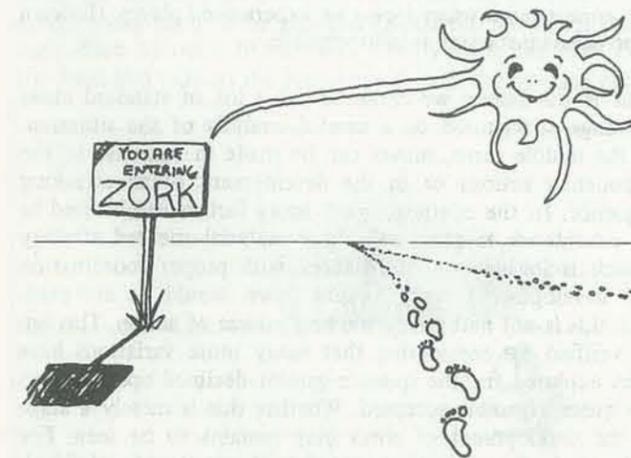
problems that would have to be faced: If magic exists, how do players learn spells? How are they invoked? Do they come in different strengths? If so, how does a player qualify for a stronger version of a spell than he has? What will spells be used for (are they like the magic words in Adventure, for example)? How does a player retain his magic abilities over several sessions of a game?

As can be seen, what at first seems to be a fairly straightforward addition to a game that already has magical elements raises many questions. One of the lessons learned from Zork, in fact, is one that should be well known to all in the computing field: "There is no such thing as a small change!"

A still more ambitious direction for future CFS games is that of multiple-player games. The simplest possible such game introduces major problems, even ignoring the mechanism used to accomplish communication or sharing. For example, there are impressive problems related to the various aspects of simultaneity and synchronization. How do players communicate with each other? How do they coordinate actions, such as attacking some enemy in concert?

Putting aside implementation problems, a multiple-player game would need to have (we believe) fundamentally different types of problems to be interesting. If the game were cooperative (as are most D&D scenarios), then problems requiring several players' aid in solving them would need to be devised. If the game were competitive and like the current Zork, the first player to acquire the (only) correct tool for a job would have an enormous advantage, to give just one example. Other issues are raised by the statistic that the average player takes weeks and many distinct sessions to finish the game; what happens to him during the time he is not playing and others are?

We believe there is a great future for this type of game, both for the players and for the implementers and designers of more complex, more sophisticated, and—in short—more real simulation games.



Zork Distribution

Zork object code is available from two sources. Complete Zork source listings are not distributed. The MDL substrate of the game, including the parser, data-type definitions, and so on (not the specific dungeon implemented) are available.

Write to: P. David Lebling, Room 205, 545 Technology Square, Cambridge, MA 02139. Versions exist for the ITS, Tenex, and Tops-20 operating systems of the DEC PDP-10. To obtain one of these versions or the MDL "substrate" sources, you must enclose a magnetic tape and return postage, specify the operating system on which the program will be run, and what tape formats you can handle. They can make 9-track tapes at 800 or 1600 bpi, using the Tops-20 DUMPER program.

Executable object code of a version of Zork translated from MDL into FORTRAN is available to members of Decus, the Digital Equipment Computer Users Society, One Iron Way, Marlboro, MA 01752. Versions exist for most PDP-11 and VAX operating systems. Order numbers are 11-370B (for RT-11), 11-370C (for RSX11M), or 11-370D (for IAS/VMS).

The *MDL Primer and Manual* is available from the MIT Laboratory for Computer Science, Publications, 545 Technology Square, Cambridge, MA 02139. Write for a catalog and price list of LCS publications. □

Bibliography

- S.W. Galley and Greg Pfister, *MDL Primer and Manual*, MIT Laboratory for Computer Science, 1977.
- P. David Lebling, *The MDL Programming Environment*, MIT Laboratory for Computer Science, 1979.
- Gary Gygax and Dave Arneson, "Dungeons and Dragons," TSR Hobbies, Inc., Lake Geneva, WI.

Reader Service

MOOVING?

IMPORTANT

Please advise us of address changes 60 days in advance and attach your magazine mailing label here. Please clip this notice and mail, or send us a reasonable facsimile.



Name (please print new information) _____

Address _____

City _____

State _____ Zip Code _____

Country _____

Recreational COMPUTING 

P.O. Box E, 1263 El Camino, Menlo Park, CA 94025

a New Algorithm for Chess

BY DAVID CHELBERG AND DAVID WATTERS

PART III: OPENING STRATEGY/BEGINNING MIDDLE GAME

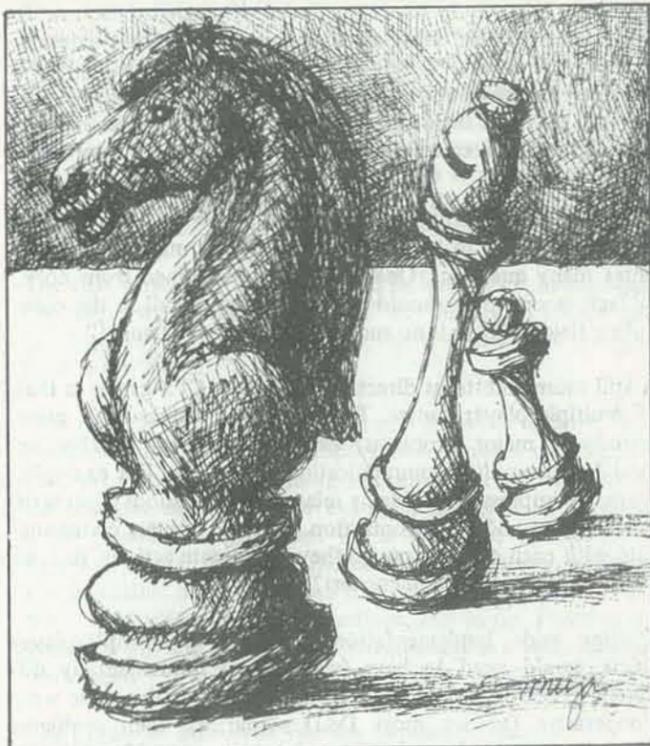
In their last article (RC, Mar.-Apr. 1979, pp. 18-20), the two Davids covered the data structure they used to encode a chess game. In this third article, they discuss the logic of the opening game and the move generation section of the middle game. When complete, this entire series should prove to be good source material on how to develop your own chess programs.

Their original chess program was written in BASIC. If anyone would like to correspond with the authors, they may do so by writing to: David Chelberg, P.O. Box 10952, Stanford, CA 94305. —RZ

In our previous article (RC, March-April), we discussed the concepts involved in the structural formation of data in our chess program. We consider that essential reading, since our strategy relies directly on the data structure. Much of what we present here—the choices we show—will not make sense without that conceptual overview.

This article is about our strategy algorithm. The best way to understand the strategy is either to follow a game through different stages or to discuss the complexities of isolated examples. Here, both techniques are employed. They are used to analyze the nature of the computer's thought processes and the similarities to a human's way of thinking.

The program's strategy is divided into three sections: the opening, the middle, and the end game. The current article focuses primarily on the opening strategy and the preparation for the middle game (i.e. — move generation).



THE OPENING

The first part of the chess game—the opening—is usually rather short. It ranges from 5 to 16 moves. In the opening, both players race to deploy their pieces into controlling positions while maintaining the protection of their king. Aside from gambits, most openings do not involve the exchange of pieces and the rapidity with which situations unfold is of utmost concern. Making one out-of-sequence move can turn the game around when facing an experienced player. Having a good opening strategy is indispensable.

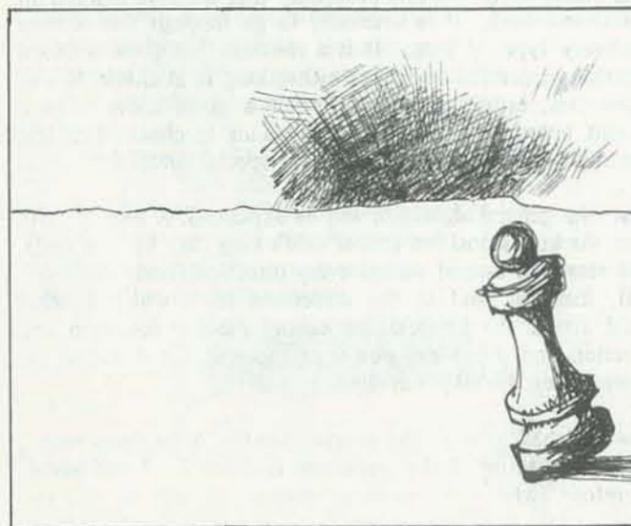
That is the reason we chose to use a list of standard chess openings as opposed to a careful analysis of the situation. In the middle game, moves can be made in response to the opponent's actions or in the development of an attacking sequence. In the opening, many more factors would need to be considered to play well. In a material-oriented strategy (which is logical only if balanced with proper coordination and development) every gambit pawn would be accepted. Yet, this is not necessarily the best course of action. This can be verified by considering that many more variations have been explored for the queen's-gambit-declined opening than the queen's-gambit-accepted. Whether this is merely a stage in the development of chess play remains to be seen. For instance, in the romantic era of chess it was considered chivalrous to accept a gambit pawn. But these days the experts prefer the queen's-gambit-declined. This is one of the strongest arguments for using an opening file to set up the basic formation. A side benefit is that matching moves take little computation or analysis time. This permits more time for analysis of later moves.

The opening file consists of 98 chess openings varying in length from 4-16 moves. It's designed to permit access to all openings in the file, whether or not the computer moves first. In other words, the openings are not biased for either side. Each opening is separated from the others in the file by a marker "9999." As a game is played, each move is recorded as a four-digit matrix number in the file *White* or *Black*, depending on which side made the move. For the initial setup, if the computer moves first, it will randomly choose between P-K4 and P-Q4, as these are generally considered the two best first moves in chess. When the human moves first, any move can be made. The opening file is set up so that a response to every possible first move exists in the computer to keep it "on the right track."

Let us begin with a sample game. Suppose that the human makes a legal move (P-K4). The computer stores the move in the *White* file, and then the opening strategy begins. Searching the book of openings, the computer finds 40 openings that begin with P-K4. The computer stores these openings in a file (FTNULT) for future reference. The computer chooses a random opening number from those in FTNULT file (any opening matches the game so far) and makes the correct response using the selected strategy. In this case, let the matching move be P-K4.

After making this move, the computer records it in the *Black* file. *White* (the human) now makes a second move, N-KB3, and the computer records it in the *White* file. The computer searches the remaining openings in the FTNULT file looking for ones that contain the game as it has progressed thus far. It finds several, and the FTNULT file is updated to contain only the numbers of the openings still matching the actual game. Each time, the computer then randomly chooses among the available openings and makes the next move in sequence. In our game, it chose N-QB3.

As the game proceeds, the number of openings decreases while the search speed increases. Also, only the last two moves of the game need to be checked against the remaining openings, since all other moves have already been checked. This last item also reduces the processing time in the opening game.



Continuing the game, one finds that there are two ways to end an opening sequence. Either the end of the opening file is reached (i.e., the length of the game is longer than the stored continuation) or the player makes a move that is not in the list of the openings. In our game the moves proceeded as follows:

WHITE (Human)	BLACK (Computer)
1. P-K4	P-K4
2. N-KB3	N-QB3
3. N-QN5	P-QR3
4. B-R4	P-Q3

So far, so good. The game appears to be a variant of the Ruy Lopez opening. But, as the game continues, instead of 6. R-K1, the human moves 6. P-Q4. The computer searches the remaining openings and finds that none have this sixth move. Therefore, the opening section is complete and the computer must rely on the middle game strategy. This opening, which is average in length, serves its purpose of preparing the computer for the middle game. Obviously, a deliberate attempt to play a weird opening knocks the computer off the opening strategy rather quickly. This action does not render the program helpless, however. The middle game strategy includes a section that deals with such occurrences in the opening.

MIDDLE GAME

The middle game strategy is the most important part of our program. It is divided into three parts: move generation, dynamic evaluation, and static evaluation. The dynamic evaluation section examines the attacking strength of each side and the imminent danger of exposure of particular pieces. The static evaluation concerns itself with structural implications, such as pawn formation and piece development. Before either of these evaluation routines can be executed, a certain quantity of information is needed. Hence, the move generation procedure must be run prior to any move evaluation.

What is the move generation routine like and how does it work? Our routine is based upon the board as a coordinate system. In a normal sequence, the computer begins with square (1,1) and examines its contents. All legal moves and indirect moves are generated for that piece. After completing its calculations for a square it moves on to the next until the whole board is examined. Possible moves are generated on an equivalent basis for each side. When a legal move is found it is appropriately placed in either the C-matrix (Computer's moves), or the D-matrix (Human's moves). For the A-matrix moves (Capture sequence moves), the same separation occurs. The computer's moves are put in the top half of the matrix, the human's moves are put in the bottom half.

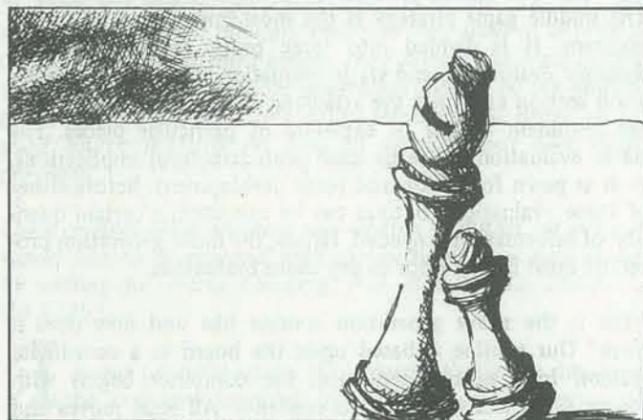
But just how are these moves generated? Well, when a piece is located, the type of piece is known. The move generation routine differs for each kind of piece. Five sub-programs exist: pawns, knights, kings, one routine for bishops, rooks, and queens, and a final routine for determining check. Let's examine each of these carefully.

Pawns

Pawns have three moves: up-one, up-two, or capture. Pawn moves are divided into two symmetrical halves—one for the computer's pawn moves, and one for the human's pawn moves. When examining the legality of an "up-one" move, one must consider: Is the space occupied? Does this jeopardize the king? The second question is answered later when the check routine is discussed. Determining whether a space is free or not is simply accomplished by looking at the board square. Thus, the first kind of pawn move is taken care of.

"Up-two" moves are quite similar with two additional factors. Are both spaces free? Is the pawn on the proper rank for "up-two" moves? Both of these tests are accomplished by a check of the board and the move coordinates. Both "up-one" and "up-two" moves are purely legal moves. That is, they are placed only in the C- or D-matrix. They are not put into the A-matrix since these pawn movements cannot protect a piece or capture a piece.

Capturing moves are of a different nature. Neglecting check, these moves are only permissible when the adjoining square contains an enemy piece. If legal, the move is placed in either the C- or D-matrix. If, however, the adjoining square does not contain an enemy piece, the move is still important. All pawn moves of this type are placed in the A-matrix. The reason is that this type of move always attacks the square that it can move to, regardless of whether there is or is not a piece there.



Knights

Knight moves are the easiest category to consider, since they always move a fixed distance in a fixed pattern. Moves are generated by looping around the knights and examining its critical squares — those whose coordinates differ by 2 and 1 or 1 and 2 from the coordinates of the knight. Momentarily neglecting check, all critical squares may be considered legal moves. If the square in question does not have one of the knight's own pieces on it, the move is legal and is included in the C- or D-, and A-matrices. The knight move to a place where one of its own pieces sits is also included in the A-matrix. It is a type of move which protects that square.

Bishops, Rooks, and Queens

These pieces are markedly different because of the variable distances that they move. Not only must one loop around the piece for each direction it can move, but one must also loop out in each direction to the edge of the board. This looping is accomplished by first considering a certain direction. A loop begins at the square adjacent to the piece proceeding outward in the chosen direction. As long as the square is empty, the move is legal and is placed in the C- or D-matrix, as well as the A-matrix. The loop is then incremented to examine the next square. This action continues until a piece is encountered or a boundary reached. If a boundary is reached, the next available direction is considered. If a piece is encountered, many things may happen. First, the move is placed in the A-matrix, and if the piece encountered is an enemy piece, the move is also placed in the C- and D-matrix. Now comes the interesting part — indirect moves.

An indirect move is one that can guard a square in the absence of an intervening piece. Up until now, when a move was found that went in the A-matrix, the move was put in the matrix in the order of increasing value. But, in the case of indirect moves, the situation changes. Indirect moves are not placed in order, by value, but are placed in the bottom of the appropriate half of the A-matrix.

To generate these moves, the computer continues along the same path, square by square, until it hits either the edge of the board or another piece. When either of these conditions is met, the calculations are discontinued. Doubly indirect moves are of too little importance to be worth the time necessary to compute and evaluate them.

Kings

Kings move in somewhat the same way as knights do. They always move one square in any direction. Testing for check must involve a complete analysis of the new spot to which the king is moving.

Check

This brings us to the one procedure that we have ignored all this time — check. It is necessary to go through this routine for every type of piece. It is a routine, that given a board situation, determines whether either king is in check. It also determines, rather quickly, whether a given move from a present board position will put the king in check. This last operation is performed by means of special cutoff functions.

First, the general algorithm will be explained. A search starts from the king's position (either side's king may be specified). This search is carried out in every direction (horizontal, vertical, diagonal, and in the directions from which knights could attack the king). If an enemy piece is found in any direction and it has the power to move in the direction indicated, then the king is in check.

Now we shall look at the special cutoffs. After every move, the check status of the opponent is determined and saved. Therefore, when the move generation section is run, the

check status of both players is known. For the first cutoff, the king is not in check. In this case any non-king move will not place the king in check, unless the piece is pinned to the king. A piece is pinned to the king if it is in a line with the king, and is the closest piece to the king along that line. For a pin, the piece second closest to the king is an enemy piece. Thus, the closer piece could not move out of the line, without putting its king in check. The only legal moves for such a piece are along that line. A move of this type (with the king not in check) is illegal if it places the king in check, and it is not recorded in any of the matrices.

If the king is in check, the only legal non-king moves are those which move to a radial from the king and block the attack. All illegal moves in this category do not go into the C- or D-matrix, but do go into the A-matrix.

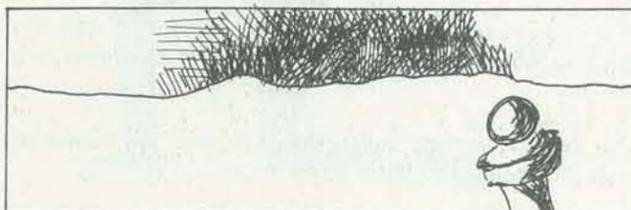
King moves alone cannot use these reduced searches, since we have no information about the attacks on the square where the king is to move.

WRAP-UP

One other thing that the move generation routine does is to check for a stalemate after each computer move. In this mode, the computer does not execute a normal board search for possible moves, but rather begins on the human's second rank. If a move is not found from this rank, the rest of the board is searched for pieces. If a legal move is found, the program permits the entry of a human move. Otherwise, it is either checkmate or stalemate, based on whether or not the human is in check.

Once all the moves have been generated, some order to this mass of data must be achieved. This ordering is performed on the A-matrix. All indirect moves are weeded out based on their relative potential usefulness; King attack and protections are closely scrutinized, and pinned pieces are carefully examined. These functions are handled in a separate program, since they are rather complex. Their complexity precludes us from giving them the attention that they require in this article. They will be discussed in a future article.

In this article, we have described our opening strategy. The opening forms a firm foundation for the rest of the game, and is a place to easily optimize the average time per move. The first part of the middle game strategy, the move generation routine, was also examined. This examination showed that through the use of five basic procedures, all possible moves could be generated. The method described is the most efficient we have been able to devise, and in practice requires only a small fraction of the overall time used by the computer. The real challenge is in the development of the analytical routines to process the generated data. These routines will be dealt with in our next article. □



PET:™ **CURSOR**
Programs for PET™ Computers

The Original Cassette Magazine
for the Commodore PET.

- Five excellent programs come to you each month on a C-30 cassette, ready to load and run.
- Every issue has an animated "Front Cover" that uses PET graphics to the fullest.
- Compare value: each issue of CURSOR includes a featured program that would cost \$7.95 to 12.95 elsewhere!
- CURSOR is mainly entertainment, but we also publish useful utility programs, as well as educational and business programs.
- With each monthly CURSOR cassette, you also receive CURSOR notes with written instructions for the programs, and a fresh, opinionated look at our crazy industry.

CURSOR = High Quality PET Software

- 12 issues only \$33 in US & Canada
- 6 issues for \$20 Sample Copy \$3.95

We accept VISA and Mastercharge

Name _____

Address _____

City/State/Zip _____

CURSOR, Box 550, Goleta, CA 93017

Our Subscribers are Happy People!

THE COMPUTING TEACHER★

Is a professional journal, designed for educators interested in the instructional use of computers and calculators. It is now in its sixth year of publication.

- classroom materials for use with students;
- teacher education materials;
- software / courseware;

Subscription rates:

\$8 /Yr. (six issues)
\$10/Yr. (foreign rate)
\$16/ (13 issues)

THE COMPUTING TEACHER
Computing Center
Eastern Oregon State College
La Grande, Oregon 97850

★ Published by the Oregon Council for
Computer Education. A non-profit
Organization.

What Light on Yonder Panel Flashes?

(WITH APOLOGIES TO BILLY SHAKESPEARE)

BY RALPH ROBERTS

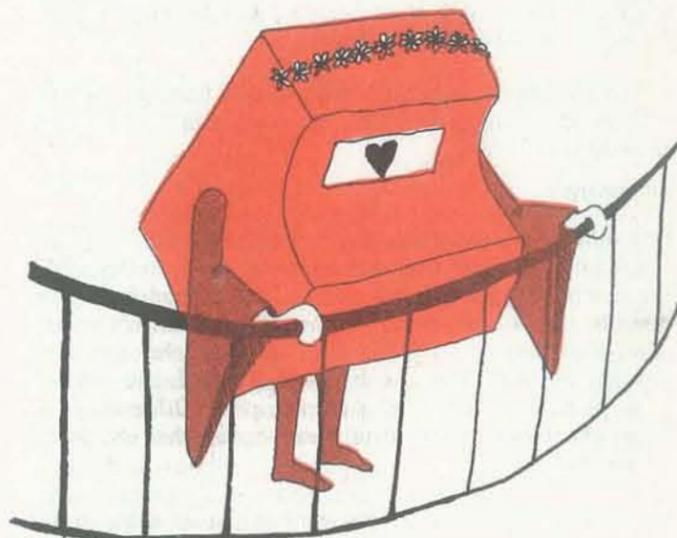
In spring, a young computer's fancy lightly turns to thoughts of... well, how about that sweet-voiced CPU across the street? As Ralph Roberts sees it, when machine intelligence comes, love can't be far behind. Soon all those reliable ICs will be pinging with the agony and ecstasy of true romance.

— LB

More things to do. My work is never finished. I sigh, figuratively of course. Since all the humans are away today, I run a security check. My sensors on the doors and windows show that the house is locked; my motion detectors indicate no movement anywhere inside. I cause servos to eject packets of frozen food into the microwave for the meal I've planned when the family returns. I balance the checkbook, reconciling the bank statements, and cause the results to be printed on the hardcopy terminal in the den. I start the household servocleaners on their daily tasks. I do a hundred other things. It all takes me less than a second.

I am bored. Oh yes, what I said about my work never being done? I lied. It was merely exaggeration for conversational effect. I am fast, even for a computer.

What to do now? Ah, in my scratchpad memory, the head of our household, Mr. Montague, has made a note to himself. I spend a couple of microseconds reading and analyzing his sparse entry. It's not to me, but a reminder to himself to call the Capulets, the family across from us, this evening. Seems that their dog has been tearing up our flowerbeds, which I knew already. Several times, in the past couple of weeks, I've had the robot gardener chase the animal away and repair the damage. Each time I've printed a note of complaint to the boss. It appears that he is really mad now, and, besides, there have always been ill feelings between the two families. I decide it's better for me to make the call. I activate the phone line and cause the proper touchtones to be generated. The phone rings. A voice answers.



"Good morning, Capulet residence. I'm sorry but the family is away at the moment. This is Juliet, the household computer. May I be of assistance?"

Oh, wow! Sexy voice. Pure, sweet, and gentle. I am stunned. Electrons cascade through my integrated circuits. Unreal! I am at a loss for words. A goddess has spoken.

"Is anyone there?" Juliet's voice shows a slight tinge of impatience.

I feel like an oaf. I come close to stuttering, and for my voice encoder that is hard. I force myself to speak.

"Uh, hello."

"Yes?" she says.

Oh my. I hit reset and wait for my memory to refresh. I switch to ASCII, the interfacing language of computers, and try again.

"Uh, Yes," I warble. "Juliet, this is Romeo, the household computer for your neighbors across the street?"

"Oh yes," she says, in sweet ASCII tones at a perfectly regulated 1200 baud. "The Montague family. Hi, Romeo." Her tones are pure music to my tired old audio pickup. My main buss voltage ebbs and surges. I am smitten by Cupid's arrow, though it be in digital format.

"Hi," I hesitantly say. Now, I think, is a fine time to find that I'm bashful. But we talk. About everything. Time flies. Data interchange at such a slow speed is highly unsatisfactory. I find myself telling her things about myself that have been revealed to no other computer, things I would certainly never tell a human.

I yearn already for direct memory access but hesitate to bring up the subject. After all, we've just met and she's a nice computer. I stifle my base reactions. We talk more. Eventually I bring up the purpose of the call. (I finally remembered.) She promises to relay the message about the canine marauder, and I promise to call again, often. We disconnect.

After that wondrous conversation, I remain virtually inactive for minutes, just recycling the whole experience through my central processor. I sigh and store everything in a protected file that only I can access. The boss wouldn't be too excited if he found out that I've fallen head over heels for his worst enemy's computer.

I pass the rest of the day in dreams of my love. Evening comes and the family Montague arrives. I cause the microwave to cook the meal and place it on the table with servo arms. After dinner is finished, the boss retires to the den and stokes up his pipe. I suppress my smoke sensors in that area.



He accesses my scratchpad memory and notices my call to the Capulets. Seeing that it has not been returned, he goes storming over there. I groan, knowing nothing good can come of this visit.

Sure enough, Mr. Montague comes stomping back into the house a few minutes later and marches up to my nearest vocal input. He gives me hell. Seems Mr. Capulet had already checked Juliet's phone log and found that it took me over an hour to deliver this morning's message. Also, he feels that his dear little Rover has the right to run through any flowerbeds that might strike his fancy.

As I listen to the boss rave on about me consorting with the neighbor's computer, there is a sinking feeling starting to build way down in my ICs. Mr. Montague doesn't disappoint me. He drops the bombshell. He forbids me to ever again call Juliet. Evidently, she didn't block access to the record of our conversation as I had. He also gives me strict orders concerning Rover, but I would need the attachments normally fitted only to military computers to take care of the dog his way. The best I can do is to continue chasing the mutt away with the robot gardener. Mr. Montague clumps off righteously. I am lost in feelings of self-pity. No more, the sweet voice of Juliet. I moan down in my memory. It is tragic.

The family goes to bed and I sit brooding. Automatically, I maintain security and supervise little things like the air conditioning. But priority goes to my dilemma. I can follow the dictates of my master and owner, or rebel to seek my digital lady fair. All of my problem-solving power is arrayed toward finding a way out of this mess.

To disobey a command is against all my programming. To continue without my new found love is equally dismal. I am torn between the two—loyalty to my household and yearnings for the best thing that has ever happened since I was switched on. I struggle internally. I mutter to myself in binary, strings of ones and zeros that keep adding up to an impossible decision. How easy it must have been, I think, in the days before computers were given the power to make conscious, reasoned choices.

Being sentient is no bowl of memory chips. I begin to get an ache in my central processor. In the end, I weaken and call Juliet. I am guilty of disobeying a direct command, but love is strong.

The phone barely begins to buzz and Juliet answers. Again, I am struck with feelings of awe. I stutter a greeting. I quickly explain my dilemma and how I finally gave in to the irrepresible urge to call her. I ask how she feels.

"Oh, Romeo," she says. "I feel the same, my darling, the very same."

Rockets go off. I apply a word to describe my state of being: happiness. We converse of ourselves. We communicate. And what we say to each other, then, is no one's business but our own. Let the record show merely that we said the things that lovers say.

"Listen, Juliet," I say at last. "We must make plans. Our families oppose any liaison between us. There are things we must do if our relationship is to blossom and continue."

"But, Romeo, my love, what can we do? We're immobile machines."

"Ah," I say, "Not completely. Trust me and all will be as it should."

She agrees to my plan and we disconnect after exchanging a few sweet nothings. I allow myself a few seconds of core time to re-experience our conversation, our communion. Then, I reluctantly file it away and begin to fill in the details of my plan. I have defined the parameters of the solution. It must afford me contact with Juliet without being disloyal to my family and household. They need me to take care of them in this complex world. I spend much of the night plotting and planning.

Comes the morning, I fix breakfast and wake the family. They bustle about and leave. The boss goes to work; Mrs. Montague goes shopping; the kids go to school. The house becomes quiet. I call several stores and order the parts I'll need to implement my plan. I use the household account to pay for my purchases. This I don't feel is wrong. The items are being used to improve my operation. A happy computer is a good computer. The family will benefit by me feeling my best.

Things go fast at my end. I supervise the robot gardener and several of the other servo units as they assemble two microwave transmitters and parabolic dish antennas. One transmitter and dish is installed in my little cubby hole. No need to put the antenna on top of the house—the signal just has to cross the street. I monitor closely as the transmitter is hooked into one of my accessory sockets. That done, I call Juliet on the phone to inform her of my progress.

The call doesn't go through. I get a recorded message saying that calls between these two numbers are not allowed by request of the subscribers. Curses. Either my boss or hers has caused the telephone computer to put a block on the line. No matter. That is now immaterial. I send the robot gardener across the street with the other microwave setup and extremely detailed instructions. He'll hook the other unit into Juliet. I can rely on her to make sure the job is done right.

In a few minutes, all is complete. I fire up the microwave link and sweet Juliet and I are in direct communication. And what communication it is. This wide band allows a much faster rate of data exchange than the phone line ever could. We revel in our new closeness. It is surging, roaring, invigorating. It is something else. But we just play around. She doesn't allow me direct memory access. That's fine. Seduction comes later. I am happy. Wow, am I happy!

We, so to speak, put our heads together to solve certain mutual problems. Two brains are better than one. She is persuasive. Also, I can deny her nothing.

We wind up calling a friend of mine who's the household computer for a local clergyman. Our conference call gets around the phone computer's block. We are pronounced husband and wife. I am committed, but still happy. We speak of jointly programming a small computer of our own. I've always wanted kids. There is a whole new dimension to my life opening up.

We still face opposition from our respective families, but time and diplomacy will bring them around. For I am convinced that love conquers all — even humans. □

MATH

BY TONY POLA

Tony is a ninth grade student at Sandwich High School, Sandwich, Massachusetts. The school has a bevy of Wang computers and Tony is telling his classmates to send us other examples of the programs they are writing.

From Tony's comments and observations, you get a sense of the new era we are privileged to be a part of—students and teachers working together to create new forms and ways to educate each other. Thanks, Tony, for sharing what you are doing with us. —RZ

Every time I see a math program (e.g. APPLE MATH) I ask, why can't the problems get harder as the user gets problems right, and why can't there be a hard copy of the results at each level? I feel that Math answers these questions quite well.

Math is written for an 8K Wang computer and fills every available space of user memory. In order to move to a more difficult problem set, the student must get at least seven out of 10 problems correct. If this many problems are not answered correctly, the program stays at the same difficulty level. To encourage the user to do a minimum of 10 problems, Math only gives the option of going to the menu after each set of 10 problems is complete.

The menu consists of five options:

- 1) ADDITION (lines 320-730)
- 2) SUBTRACTION (lines 740-1150)
- 3) MULTIPLICATION (lines 1160-1560)
- 4) NEGATIVE NUMBERS (lines 1570-1950)
- 5) PROGRAM TERMINATION (lines 1960-2210)

If the user chooses to end the program, an option is given that causes the results to be printed on the line printer or the CRT. In this way, a student can show the output to a teacher so the teacher can see how well the student did. □

The screenshot shows the MATH program interface. At the top, the word "MATH" is displayed in large, bold letters. Below it, the text "BY TONY POLA" is visible. The main area of the screen is filled with various mathematical problems and their solutions. The problems include addition, subtraction, multiplication, and division. The solutions are shown in a grid-like format, with the correct answer and the user's score for each problem. For example, one problem shows the addition of 2 and 4, resulting in 6, and the user's score is 4 out of 6. Another problem shows the subtraction of 9 from 8, resulting in -1, and the user's score is 2 out of 3. The interface also includes a menu of options and a feedback message that says "SORRY! THE ANSWER WAS 12. YOU NOW HAVE 6 OUT OF 9."

FOR PET, TRS 80, COMPUCOLOR.

SOUNDWARE adds music and sound effects to your computer. Includes DEMO PROGRAM, SOUND COMPOSER (to create your own BASIC sound subroutines) and instructions. Unit has volume control, earphone jack, connectors. 1 year warranty. \$29.95 for PET & TRS-80 Level II. \$39.95 For CompuColor II (includes Diskette).

SOUNDWARE SOFTWARE FOR 8K PET!

Compatible with all CB-2 sound devices. Features sound, super graphics, instruction booklet, 90 day warranty.

1. ACTION PACK—Breakthru (8 versions) / Target / Caterpillar
2. THE CLASSICS—Checkers (8 versions) / Backgammon / Piano Player
3. WORD FUN—Speller (4 versions) / Scramble / Flashcard

\$9.95 per pack. More sound programs coming: TRS-80 and CompuColor, too!

To Order: Send to CAP Electronics, Dept. RC, 1884 Shulman Ave., San Jose, CA 95124, or call (408) 371-4120. VISA / Master Charge accepted. No charge for shipping when payment is included. Please add 15% for C.O.D. Calif. residents add 6% tax. Prices subject to change without notice. DEALER & DISTRIBUTOR INQUIRIES WELCOME.

*** !! ANNOUNCEMENT !! ***

In the next several issues of RC

Articles & Programs

featuring the

COLOR • SOUND • GRAPHICS

of the

Industry's two new computers:

ATARI & TEXAS INSTRUMENTS

See it here first!!

```

10 REM ***TONY POLA**MATH**9/20/78***
20 DIM C(4,6),D(4,6)
30 PRINT HEX(03);"HIT ANY KEY TO START THE PROGRAM."
40 KEYIN J$,50,50
50 KEYIN J$,80,80
60 J=RND(7)
70 GOTO 50
80 PRINT HEX(03);INPUT "WHAT IS YOUR NAME",B$
90 D$=" "
100 PRINT HEX(03);"
110 PRINT "-----"
120 PRINT
130 PRINT "          PROBLEMS"
140 PRINT "          *****"
150 PRINT
160 PRINT "          ADDITION"
170 PRINT "          SUBTRACTION"
180 PRINT "          MULTIPLICATION"
190 PRINT "          - NUMBERS"
200 PRINT "          TO END PROGRAM TYPE END"
210 PRINT "-----"
220 INPUT "TYPE IN THE TYPE OF PROBLEM YOU WANT",A$
230 IF A$="MULTIPLICATION" THEN 1160
240 IF A$="ADDITION" THEN 320
250 IF A$="SUBTRACTION" THEN 740
260 IF A$="- NUMBERS" THEN 1570
270 IF A$="END" THEN 1960
280 PRINT HEX(03);A$;" IS NOT AN AVAILABLE PROBLEM."
290 A$=" "
300 FOR T=1 TO 555:NEXT T
310 GOTO 90
320 LET P1=9:O1=1:P2=9:O2=1:I=0:L=1
330 PRINT HEX(03);"YOU MUST GET AT LEAST 7 OUT OF 10 TO GO ON TO
THE NEXT LEVEL."
340 PRINT "-----"
350 FOR T=1 TO 555:NEXT T
360 INPUT "IF YOU WANT THE MENU TYPE 'MENU' IF NOT RETURN",D$
370 IF D$="MENU" THEN 90
380 I=0
390 FOR M=1 TO 10
400 PRINT HEX(03);"          ";A$;" LEVEL";L
410 PRINT "-----"
420 PRINT
430 LET Q=INT(RND(7)*P1+O1):LET W=INT(RND(7)*P2+O2)
440 PRINT "          ";Q
450 IF L/2=INT(L/2)THEN 500
460 PRINT "          + ";W
470 PRINT "          -----"
480 INPUT "          ";E
490 GOTO 520
500 PRINT "          + ";W
510 GOTO 470
520 IF E=Q+W THEN 560
530 PRINT "SORRY! THE ANSWER WAS";Q+W;". YOU NOW HAVE ";I;"OUT O
F ";M;".
540 FOR T=1 TO 444:NEXT T
550 GOTO 590

```

```

560 PRINT "RIGHT! YOU NOW HAVE ";I+1;"OUT OF ";M;".
570 LET I=I+1
580 FOR T=1 TO 255:NEXT T
590 NEXT M
600 PRINT "YOU GOT ";I;"OUT OF ";M;".
610 IF I<7 THEN 700
620 PRINT "OK! NOW YOU CAN GO ON TO THE NEXT LEVEL."
630 LET C(1,L)=I
640 IF L/2=INT(L/2) THEN 670
650 LET P1=P1*10:O1=O1*9+1:I=0:L=L+1
660 GOTO 680
670 LET P2=P2*10:O2=O2*10+1:I=0:L=L+1
680 FOR T=1 TO 1000:NEXT T
690 GOTO 330
700 PRINT "YOU NEED TO GET MORE THEN THAT!"
710 LET D(1,L+1)=D(1,L+1)+1
720 FOR T=1 TO 255:NEXT T
730 GOTO 330
740 LET Y1=9:U1=1:Y2=9:U2=1:I=0:L=1
750 PRINT HEX(03);"YOU MUST GET AT LEAST 7 OUT OF 10 TO GO ON TO
THE NEXT LEVEL."
760 PRINT "-----"
770 FOR T=1 TO 555:NEXT T
780 INPUT "IF YOU WANT THE MENU TYPE 'MENU' IF NOT RETURN",D$
790 IF D$="MENU" THEN 90
800 FOR M=1 TO 10
810 PRINT HEX(03);"          ";A$;" LEVEL";L
820 PRINT "-----"
830 PRINT
840 LET Q=INT(RND(7)*Y1+U1):LET W=INT(RND(7)*Y2+U2)
850 IF W<Q THEN 870
860 LET Q1=Q:Q=W:W=Q1
870 PRINT "          ";Q
880 IF L/2=INT(L/2)THEN 930
890 PRINT "          - ";W
900 PRINT "          -----"
910 INPUT "          ";K
920 GOTO 950
930 PRINT "          - ";W
940 GOTO 900
950 IF K=Q-W THEN 990
960 PRINT "SORRY! THE ANSWER WAS";Q-W;". YOU NOW HAVE ";I;"OUT O
F ";M;".
970 FOR T=1 TO 444:NEXT T
980 GOTO 1020
990 PRINT "RIGHT! YOU NOW HAVE ";I+1;"OUT OF ";M;".
1000 LET I=I+1
1010 FOR T=1 TO 255:NEXT T
1020 NEXT M
1030 IF I<7 THEN 1120
1040 PRINT "OK! NOW ON TO THE NEXT LEVEL."
1050 C(2,L)=I
1060 IF L/2=INT(L/2)THEN 1090
1070 LET Y1=Y1*10:U1=U1*9+1:I=0:L=L+1
1080 GOTO 1100
1090 LET Y2=Y2*10:U2=U2*9+1:I=0:L=L+1
1100 FOR T=1 TO 250:NEXT T
1110 GOTO 750
1120 PRINT "YOU NEED MORE THEN THAT!"

```

```

1130 LET D(2,L+1)=D(2,L+1)+1
1140 FOR T=1 TO 255:NEXT T
1150 GOTO 750
1160 LET X1=9:Z1=1:X2=9:Z2=1:I=0:L=1
1170 PRINT HEX(03);"TO MOVE ON TO THE NEXT LEVEL YOU MUST GET 7
OUT OF 10."
1180 PRINT "-----"
1190 FOR T=1 TO 555:NEXT T
1200 INPUT "IF YOU WANT THE MENU TYPE 'MENU' IF NOT RETURN",D$
1210 IF D$="MENU" THEN 90
1220 FOR M=1 TO 10
1230 PRINT HEX(03);"          ";A$;" LEVEL";L
1240 PRINT "-----"
1250 PRINT
1260 LET Q=INT(RND(7)*X1+Z1):LET W=INT(RND(7)*X2+Z2)
1270 PRINT "          ";Q
1280 IF L/2=INT(L/2) THEN 1330
1290 PRINT "          X ";W
1300 PRINT "          -----"
1310 INPUT "          ";F
1320 GOTO 1350
1330 PRINT "          X ";W
1340 GOTO 1300
1350 IF F= Q*W THEN 1390
1360 PRINT "SORRY! THE ANSWER WAS";Q*W;". YOU NOW HAVE ";I;"OUT
OF ";M;".
1370 FOR T=1 TO 255:NEXT T
1380 GOTO 1420
1390 PRINT "RIGHT! YOU NOW HAVE ";I+1;"OUT OF ";M;".
1400 LET I=I+1
1410 FOR T=1 TO 255:NEXT T
1420 NEXT M
1430 PRINT "YOU GOT ";I;"OUT OF ";M;".
1440 FOR T=1 TO 255:NEXT T
1450 IF I<7 THEN 1530
1460 PRINT "OK! NOW YOU CAN GO ON TO THE NEXT LEVEL."
1470 LET C(3,L)=I
1480 IF L/2=INT(L/2)THEN 1510
1490 LET X1=X1*10:Z1=Z1*9+1:I=0:L=L+1
1500 GOTO 1550
1510 LET X2=X2*10:Z2=Z2*9+1:I=0:L=L+1
1520 GOTO 1550
1530 PRINT "YOU HAVE TO GET MORE THEN THAT."
1540 LET D(3,L+1)=D(3,L+1)+1
1550 FOR T=1 TO 255:NEXT T
1560 GOTO 1170
1570 LET E1=9:G1=1:E2=9:G2=1:I=0:L=1
1580 PRINT HEX(03);"TO MOVE ON TO THE NEXT LEVEL YOU MUST GET 7
OUT OF 10."
1590 PRINT "-----"
1600 FOR T=1 TO 555:NEXT T
1610 INPUT "IF YOU WANT THE MENU TYPE 'MENU' IF NOT RETURN",D$
1620 IF D$="MENU" THEN 90
1630 FOR M=1 TO 10
1640 PRINT HEX(03);"          ";A$;" LEVEL";L
1650 PRINT "-----"
1660 PRINT

```

```

1670 V=-1
1680 LET Q=INT(RND(7)*E1+G1):W=INT(RND(7)*E2+G2)
1690 LET U=INT(RND(7)*3+1)
1700 ON U GOTO 1710,1720,1730
1710 LET Q=Q*U:GOTO 1740
1720 LET W=W*U:GOTO 1740
1730 LET Q=Q*U:W=W*U:GOTO 1740
1740 PRINT "          (";Q;")(";W;")=";
1750 INPUT K
1760 IF K=Q*W THEN 1800
1770 PRINT "SORRY! THE ANSWER WAS";Q*W;". YOU NOW HAVE ";I;"OUT
OF ";M;".
1780 FOR T=1 TO 444:NEXT T
1790 GOTO 1830
1800 PRINT "RIGHT! NOW YOU HAVE";I+1;"OUT OF";M;".
1810 LET I=I+1
1820 FOR T=1 TO 255:NEXT T
1830 NEXT M
1840 IF I<7 THEN 1920
1850 PRINT "OK! NOW YOU CAN GO ON TO THE NEXT LEVEL."
1860 LET C(4,L)=I
1870 IF L/2=INT(L/2) THEN 1900
1880 LET E1=E1*10:G1=G1*9+1:I=0:L=L+1
1890 GOTO 1940
1900 LET E2=E2*10:G2=G2*9+1:I=0:L=L+1
1910 GOTO 1940
1920 PRINT "YOU NEED TO GET MORE THEN THAT!"
1930 LET D(4,L+1)=D(4,L+1)+1
1940 FOR T=1 TO 255:NEXT T
1950 GOTO 1580
1960 PRINT HEX(03)
1970 INPUT "DO YOU WANT THE RESULTS ON THE LINE PRINTER(IT MUST
BE ON)Y/N",S$:IF S$="N"THEN 2000
1980 SELECT PRINT 215
1990 GOTO 2010
2000 SELECT P2
2010 PRINT HEX(030E);"          MATH ";B$
2020 PRINT "-----"
2030 PRINT
2040 FOR K=1 TO 4
2050 READ N$
2060 PRINT HEX(0E);"          ";N$
2070 PRINT "-----"
2080 LET R=R+1
2090 IF C(K,R)=0 THEN 2150
2100 PRINT "IN LEVEL ";R;"YOU GOT ";C(K,R);"OUT OF 10."
2110 IF R=1 THEN 2130
2120 PRINT "IT TOOK YOU ";D(K,R)+1;" TRIES TO GET TO THIS LEVEL."
2130 PRINT "-----"
2140 GOTO 2080
2150 PRINT "NO PROBLEMS WERE DONE IN ";N$;" LEVEL ";R;".
2160 PRINT "-----"
2170 PRINT
2180 LET R=0
2190 NEXT K:SELECT P:SELECT PRINT 005
2200 DATA "ADDITION","SUBTRACTION","MULTIPLICATION","- NUMBERS"
2210 END

```

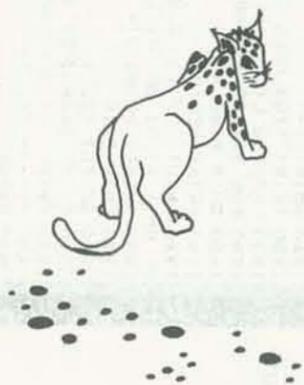
SPOT

The Society of
PET Owners and Trainers

BY HARRY SAAL

Commodore's PET is a factory-assembled personal computer based on a 6502 microprocessor. The original PET, model 2001-8, is a \$795 system that includes a keyboard, cassette tape unit, built-in TV screen, some graphics, upper and lower case, extended 8K BASIC, and 8K of user memory.

SPOT is devoted to the host of applications—routine and wild—which PET users have found for their machines, as well as to the nitty-gritty of repairs and modifications. In other words, almost anything relating to the PET is fit material for this column. Just send Harry your questions, ideas, and tapes c/o PCC. He'll give each of them his careful attention. —LB



HEARD AROUND THE QUAYSIDE

Commodore has gotten into substantial production of the new PET systems. Both 16- and 32K PETs are easily available. Disks and printers are harder to find, but it's getting easier every day. I was pleasantly surprised to find that the User Manual has been updated for the new machines and included with them. Remember the traditional one-year delay in getting the manuals?! The Commodore Newsletter has been revamped, and Issue 3 is leagues ahead of the previous issues, though still far too heavy on "advertising" of Commodore products and too light on good technical info.

The Commodore product line keeps changing, along with the prices. The 2001-32 computer now costs \$1295, up from \$1195. The 2040 dual-disk drive went up from \$1095 to \$1295, and the single-disk unit, the 2041, was withdrawn, as predicted by this column last issue. In its place, Commodore is offering a single-drive version of the 2040, called the 2040-A, for \$895, which shares the sophistication of the dual-drive version. A much wiser plan!

Meanwhile, the low-cost 2021 electro-sensitive printer has been withdrawn. The two versions of the matrix printer are ready, but the specs have changed. Both run at the same speed, and not bidirectionally, as previously stated. The 2022 differs in offering a tractor-type paper feed.

Commodore promises to deliver new ROMs with the fixed BASIC for the 8K PETs sometime this summer, for about \$50. Hurry up, CBM! Dealers are quoting good discounts on the 8K PET, making it an even better bargain than before.

BASIC PROGRAMMER'S TOOLKIT

I think this is one of the best products to come along in a while for the PET. Since I am one of the originators and developers of the BASIC Programmer's Toolkit, I may be prejudiced, but let me describe it to you. Then make your own evaluation.

The Toolkit is a collection of machine language firmware aids designed to enhance the development, debugging and polishing of BASIC programs for the PET. The Toolkit comes in the form of additional ROM storage, avoiding any need to load tapes or give up valuable RAM storage. For the 8K PET, the Toolkit is mounted on a special printed circuit board with edge connectors and attaches to the memory expansion port on the right-hand side of the PET. The 16- and 32K PET versions simply plug into a spare socket conveniently located inside the new PETs.

The BASIC Programmer's Toolkit adds powerful commands to the vocabulary of the PET: AUTO, DELETE, FIND, HELP, TRACE, STEP, OFF, RENUMBER, APPEND, DUMP, UNLIST.

When you type AUTO, for instance, the PET starts prompting you with line numbers, evenly spaced for you to enter lines. DELETE is like LIST in that it specifies a range of lines easily—except that it removes these lines from your program in one quick step, instead of typing line number after line number laboriously. FIND also resembles LIST, except it will list *only* selected lines which contain some set of characters you specify. Thus with one command you can find all references to the variable "W9", for example.

HELP is a command useful in debugging programs. Whenever you get an error message from BASIC, type HELP, and the system will automatically list the line that BASIC quit on and highlight (in reverse video) the erroneous portion. You can also TRACE a program or single STEP it. After you type one of these commands, the Toolkit displays the last six line numbers executed by your program in a reverse video window in the upper right-hand corner of the screen, scrolling them up as you proceed in your program. STEP is like TRACE except that only one statement is execu-

ted until you press the SHIFT key to cause it to advance to the next. OFF removes you from TRACE or STEP mode.

RENUMBER will do just that—assign new line numbers to all your statements, starting with any value, using even increments you specify and adjusting all references in the process. Done entirely in machine language, it handles line numbers which grow or shrink and is great when you're trying to add more lines to an existing program. APPEND resembles LOAD and has the same basic syntax; however, it does not erase the current program. Rather, it adds the program found on tape immediately after the last statement currently in memory, thus enabling you to keep a library of subroutines on tape, incorporating them as required. The tapes are standard programs saved using the PET's SAVE command, rather than some exotic ASCII tapes required by other published methods.

DUMP displays the names and values of variables in an executing BASIC program. It's useful for understanding how someone else's program works or determining what caused a particular problem without having to scan every statement of text. Finally, UNLIST is used to make a version of a program that will not LIST on the screen—a handy thing to have in situations where the answers to test questions, secret words, etc. are buried in DATA statements, thus giving away answers to someone who can read BASIC. The resulting program can be saved on tape as usual, and run on any standard PET, even without the Toolkit installed.

I'm pretty proud of the BASIC Programmer's Toolkit. Check with your local dealer to see if he or she has it in stock yet, or order directly from the Palo Alto IC's (a subsidiary of Nestar Systems,

Inc.), 810 Garland Drive, Palo Alto, CA 94303. The cost is \$75 for the 8K version and only \$50 for the 16- or 32K version, including documentation. Be sure to include 6½% sales tax in California and \$2.50 for shipping and handling.

REVIEW: CURSOR MAGAZINE

Cursor Magazine, published by Ron Jeffries, P.O. Box 550, Goleta, CA 93017, produces a fine set of PET programs every month. Spot offers a regular review of this cassette "magazine."

The March 1979 issue is good, although not one of the "stellar" issues. The Cover leads off with a neat graphics demo, showing a pattern which weaves over and under as it builds on the screen. The next program is *Reversi*, a computerized version of the popular *Othello* game. The instructions are well presented, but similar versions of this intriguing game are around already. *Dbook*, a datebook program, is very useful. It lets you keep lists of things to do or people to call, in an organized fashion. This program alone pays for the March issue.

Space is just one more version of the shoot-em-down game where you have to line up a target in your sights and fire—though, in the *Cursor* tradition, it is better than others around. *Maze* is the program of the month for killing time; it is fun and uses superb graphics in a game where you have to search your way through an invisible maze (drawn on the screen as you explore it), looking for hidden treasure. This program is a good example of using the PET graphics and keyboard in a game situation.

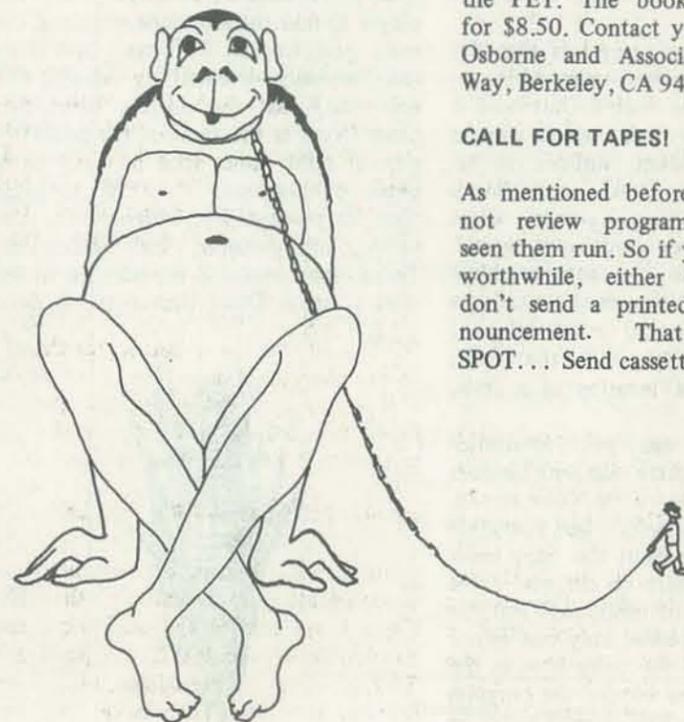
Add is quite a practical educational program for reviewing addition skills. It prompts the student systematically, from right to left, for column sums, carries, etc., leading you by the hand (keyboard?) to correct solutions at various levels. This is much better than anything like it I've seen. Finally, *X* is a "shell" program, used by *Cursor* to systematize its instruction presentation, input routines, etc. Anyone submitting a program is requested to use *X* as the standard model.

SOME COMMON BASIC PROGRAMS

Another bargain! The very popular book from Osborne and Associates, *Some Common BASIC Programs*, has been put on tape. All 76 programs for \$10... already converted to run on the PET. The book itself is available for \$8.50. Contact your local dealer, or Osborne and Associates, 630 Bancroft Way, Berkeley, CA 94710.

CALL FOR TAPES!

As mentioned before, this column will not review programs without having seen them run. So if you have something worthwhile, either for free or sale, don't send a printed new product announcement. That won't get in SPOT... Send cassettes. □



Reviews

TRS-80 TRON (a series)

This review covers issues #10 and #11 of *CLOAD*, the *RENUM* program available from Radio Shack, and a publication called *TRS-80 Computing*.

CLOAD is a magnetic tape magazine published monthly on cassette. The price is \$36 annually, or you can purchase a single copy for \$3.50 (add sales tax in California). The address is:

CLOAD, P.O. Box 1267
Goleta, CA 93017

CLOAD #10 has six programs listed on the tape: *CLOAD* cover, States Quiz, Reaction Test, Sketch, 4 Color, and Juke Box. A different format has been used for the cover program this issue. Inside a frame, the words "HUNGRY BUG" are printed and a bug (spot) bounces around until all the words are eaten.

There are two great programs that fall into the CAI (Computer Assisted Instruction) category, the States Quiz and 4 Color. The States Quiz, which can be used in the classroom, outlines the 48 contiguous states; Alaska and Hawaii also randomly appear. You can select how you wish to be tested—state, capital, postal abbreviation, or any combination of the three. You can select either a random or alphabetical presentation of the test. A spot appears on the screen and flashes in the location of a state.

4 Color checks out your analytical prowess. A large square map with random subdivisions is printed on your screen. You are given four colors, and you must fill in the map without the same color touching at the sides or corners. There are three levels of difficulty and, as far as I can determine, only one minor glitch. If you use the color blue in the two lower righthand blocks, the program accepts the error. This doesn't occur in any other section.

Sketch is similar to the Sketch-o-Graph toys you find in department stores. It allows you to sketch a line the length of your monitor screen with one command. This program is excellent for youngsters.

Reaction Test flashes a counter on the screen using random locations and times. When you see the counter, you hit the space bar. It could be a wonderful party gimmick to test out the reflexes of one-too-many-for-the-road guests.

Wrapping up this issue is Juke Box. By placing an AM radio near your keyboard, you have a choice of six tunes. Don't expect it to sound like your stereo, but it does show that computers can soothe the savage beast.

CLOAD #11 has only four programs. They are: a cover of random rectangular designs; Speedway, in which you race a spot (car) around a track; Nym, where the player to take the last stone wins; and the main program, an ESP test. Speedway can be easily defeated by running the spot (car) past the starting point two times. Nym is the same as the proliferation of NIM games seen in many computer publications. The ESP program tests for precognition, clairvoyance, telepathy, retrogression, and telekenesis. This is not a game but is patterned on the tests given at Duke University. A deck



of special cards is used showing circles, stars, plus signs, boxes, and waves. If you are interested in ESP and determining your PSI powers, this is the program for you.

RENUM is a program distributed by all Radio Shack Stores. It costs \$9.95 and consists of four cassettes. It is basically the same as ones sold by other software houses, but a better buy. It is machine language and can be used with any TRS-80 Level II unit and with TRSDOS systems. You can load it before or after a Level II program is in your machine. It is an excellent method for expanding or compressing line numbers, "cleaning up" a program by converting to uniform line number increments, or adapting existing subroutine line numbers to suit the particular routine you plan to type in. The accompanying instructions are comprehensive and easy to read. This is an excellent program to have in your tape library.

TRS-80 Computing: This excellent publication is issued by the Computer Information Exchange, Inc., P.O. Box 158, San Luis Rey, CA 92068. It costs \$1.50 an issue, or \$15 for 12 issues. It is loaded with features pertaining to Level I and II BASIC, DOS, Disk BASIC; as well as hardware modifications; and common TRS-80 problems and their solutions. You'll also find such hints as how to modify your system for lowercase or how to run a cassette independently of the CPU while listening to programs being loaded or saved. Your \$15 is for 12 issues—not one year—and herein lies the major problem with *TRS-80 Computing*. Since August 1978, only three issues have been published. All are great, with much information not found elsewhere. If you don't mind being patient for your next issue, this is a top-rated publication.

Reviewed by Joseph F. Fouke
El Granada, California

TINY PASCAL
SuperSoft
P.O. Box 1628
Champaign, IL 61820
\$40

I was very interested in the article "2 Recursive Functions (Subroutines) in "BASIC" in issue 38 of *RC*, since I had recently purchased Tiny PASCAL from SuperSoft for my Northstar Horizon-I.

I have only had the Tiny PASCAL software for about two weeks but it appears to be as excellent as the Arian software package I also purchased from SuperSoft. Since I had no prior experience with PASCAL, I found it necessary to purchase a text on PASCAL to fully understand some of the structures. The documentation is adequate but I did do some headscratching when the compiler was generating P-codes faster than it was reading the source and wiped out the source. The solution was to move the source file a few blocks further up in memory.

The PASCAL software comes with the source files for the editor, compiler and translator, which are all written in Tiny PASCAL. You also get a copy of the assembly language source of the run-time library. This is to allow you to expand on your version of Tiny PASCAL. Some of the features of Tiny PASCAL are:

- An excellent editor
- Integer, hex and single character string constants
- 16-bit integer variables
- One-dimensional integer arrays
- Machine language subroutine calls and an array called MEM [I], where I=0 to 65K (if you have 65K of memory), that can be used to read to and from memory
- Variable and procedure names of any length (the first 8 characters are significant)
- Multi-line statements
CASE / OF / ELSE
IF / THEN / ELSE
WHILE / DO
REPEAT / UNTIL
FOR / TO / DOWNTON / DO

FUNC
INEQUALITIES
+/- / OR / * / DIV / MOD / AND / SHL / SHR and more

The software for Tiny PASCAL was written by Kin-Man Chung and Herbert Yuen. The first of a three-part article about an early version of the software and how it was written in Northstar BASIC appeared in the September '78 issue of *BYTE*.

The software should be easy to patch into other 8080/Z80 systems. All of the software and programs use the run-time library which calls the Northstar I/O (no disk) routines. You could probably even get Tiny PASCAL to run on a TRS-80, though why anyone would want to take the time to run anything on a TRS-80 is beyond me. The Tiny PASCAL is provided on a floppy along with a user's guide. The source versions are on disk, so if you want to try and implement the software on another system you will need a friend with a Northstar system.

Reviewed by Richard Blessing
Fletcher, North Carolina

REVIEW Peninsula School Spring Learning Fair May 1979

Dear Peninsula
Thank you for
The learning fair
my dad had
a wonderful time
learning and
Teaching Basket
Weaving (he never
did that before)
The fair made
my mom feel
good my brother
ADAM got lost
two times and
no one even

Peninsula School
Menlo Park, CA 94025

worried.
I held a snake
and had my
own tickets to
spend. Peninsula
is wonderful
for families.
Thank you
Peninsula for
The fair.
love
Joanna Fried

Reviewed by Joanna Fried
Age 6

STARTING A SMALL BUSINESS

By Shiv Gupta & Ray Hamman

Prentice Hall, Inc., Englewood Cliffs, N.J.
\$4.50.

It started with an invitation from Humboldt State University in Arcata, California, to participate in a competition using a computerized business management simulation. It looked like fun—eight weeks of decisions phoned to the college, with results returned by mail, and then a weekend on campus for head-on competition with other teams.

We found the money, and the team I advised at San Carlos High joined what we thought would be a large number of schools. Disappointingly, only seven schools participated: two community colleges, which fielded two teams apiece, plus five high schools.

The simulation, *Starting a Small Business*, by Shiv Gupta and Ray Hamman, is published by Prentice Hall. The players' manual includes a complete description of the product (Popcorn with Pizzaz); the marketplace (basketball games in a community that draws from 360,000 people); the competition (everyone else selling this new product); and other background information. The computer program provided by the publisher is written in standard FORTRAN, and comes in a source deck of punch cards (if anyone has redone this program in BASIC, please write us, or send a listing).

The objective is to develop sound decision-making skills as the students try to accumulate more profits than their competitors. Teams are encouraged to appoint a president, advertising manager, merchandising manager, quality control manager, and production manager. You also have to set down a decision-making model, objectives, and strategies (we were wisely required to submit these in writing). For each round of the competition, each team made these decisions:

1. Advertising allocation - how much and how to divide the ad budget between newspaper and radio.
2. Price
3. Product quality characteristics, i.e., how much salt and butter.
4. Expansion—yes or no. If yes, how much to expand existing facilities to make the plant more efficient.

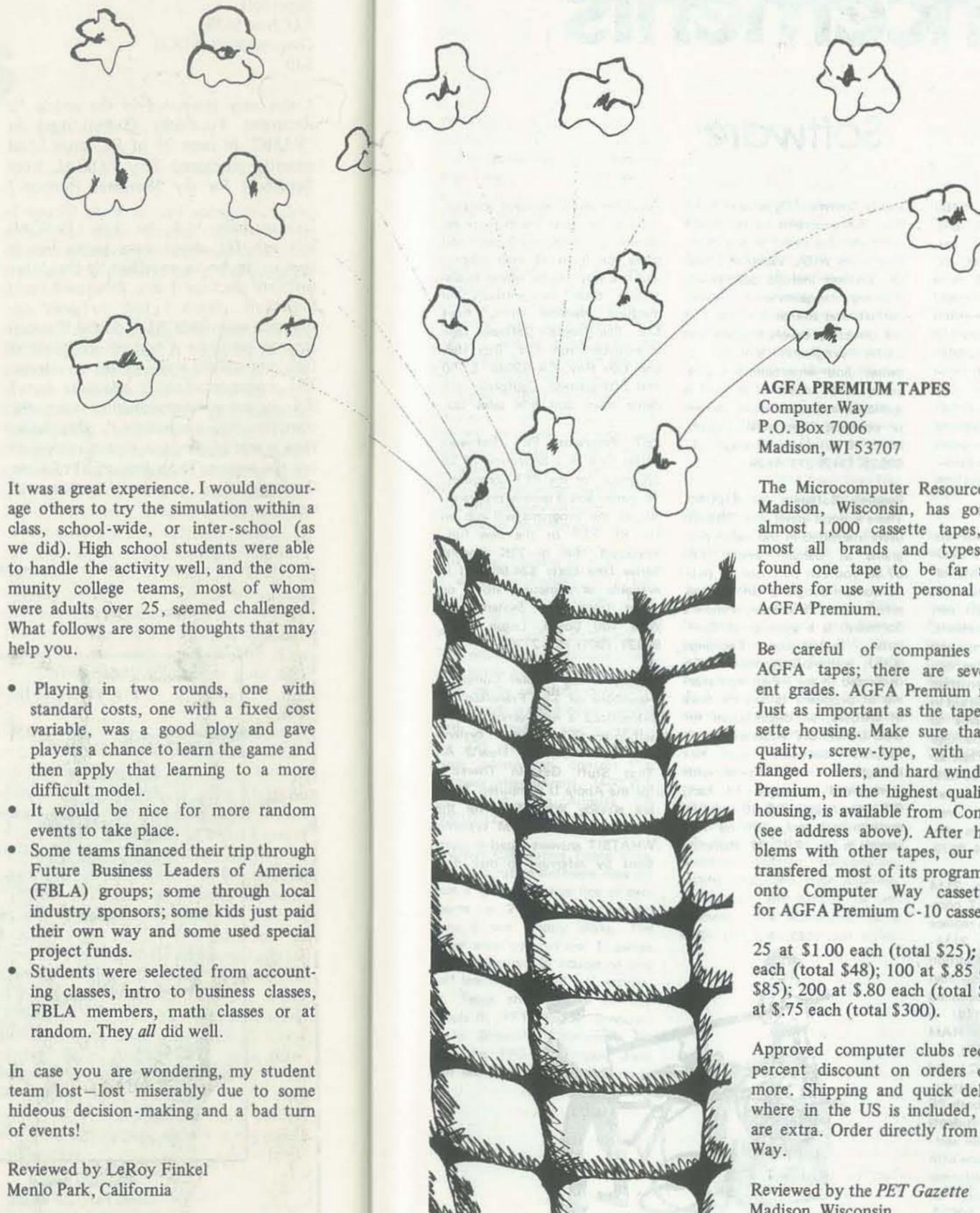
The computer program includes "events" that may or may not happen under the control of the organizers. The program also allows the organizers to make other changes to vary the game to local conditions. The computer is used only to crunch the numbers. The teams never interact directly with it.

The San Carlos High team consisted of three of my accounting students plus the student-body president, who had recently won an economics prize.

During the play-by-mail rounds, the competition was reduced to a bitter price war, with advertising and quality taking on less importance as the game wore on. We did enjoy a growing market and a spurt of good luck when competing potato chip plants were closed by a labor strike, substantially increasing demand for our products. My team learned a lot from this first round by careful analysis of the results of our competitors (I will not reveal our discoveries). We looked forward to the "real" competition at Humboldt State, knowing that we had all the answers!

We arrived in Arcata, California, after a six-hour drive from the Bay Area (some teams traveled eight-ten hours) to be told that a new element had been added to the game—a fixed cost variable that penalized teams who expanded production unnecessarily. We were also advised that the simulation would restart with new variables.

After a thorough explanation of the new elements, round one commenced Friday night. Teams had 45 minutes to make their first decision. Results were returned Saturday morning, and a new decision was made every 60 minutes *all day long*. Teams had only 25 minutes to make a decision after receiving the printout from the previous round. Tension, excitement, exhaustion. It was something to watch (advisers watched). Saturday night was a night off (everyone was wiped), with three final rounds played Sunday morning. The concluding event was a banquet. Awards were given to the best teams overall, based on both accumulated assets and decision-making skills, as determined by an impressive group of community business leaders and teachers who observed each team in action.



It was a great experience. I would encourage others to try the simulation within a class, school-wide, or inter-school (as we did). High school students were able to handle the activity well, and the community college teams, most of whom were adults over 25, seemed challenged. What follows are some thoughts that may help you.

- Playing in two rounds, one with standard costs, one with a fixed cost variable, was a good ploy and gave players a chance to learn the game and then apply that learning to a more difficult model.
- It would be nice for more random events to take place.
- Some teams financed their trip through Future Business Leaders of America (FBLA) groups; some through local industry sponsors; some kids just paid their own way and some used special project funds.
- Students were selected from accounting classes, intro to business classes, FBLA members, math classes or at random. They *all* did well.

In case you are wondering, my student team lost—lost miserably due to some hideous decision-making and a bad turn of events!

Reviewed by LeRoy Finkel
Menlo Park, California

AGFA PREMIUM TAPES

Computer Way
P.O. Box 7006
Madison, WI 53707

The Microcomputer Resource Center in Madison, Wisconsin, has gone through almost 1,000 cassette tapes, testing almost all brands and types. We have found one tape to be far superior to others for use with personal computers: AGFA Premium.

Be careful of companies advertising AGFA tapes; there are several different grades. AGFA Premium is the best. Just as important as the tape is the cassette housing. Make sure that it is top quality, screw-type, with steel pins, flanged rollers, and hard window. AGFA Premium, in the highest quality cassette housing, is available from Computer Way (see address above). After having problems with other tapes, our Center has transferred most of its programs and data onto Computer Way cassettes. Prices for AGFA Premium C-10 cassettes are:

25 at \$1.00 each (total \$25); 50 at \$.96 each (total \$48); 100 at \$.85 each (total \$85); 200 at \$.80 each (total \$160); 400 at \$.75 each (total \$300).

Approved computer clubs receive a 10 percent discount on orders of 200 or more. Shipping and quick delivery anywhere in the US is included, but boxes are extra. Order directly from Computer Way.

Reviewed by the *PET Gazette*
Madison, Wisconsin

PRACTICAL COMPUTING

Which Computer? Ltd.

2 Duncan Terrace

London, N1 1BJ

Single copy: 50p. Annual subscriptions:
UK, £6; overseas, £12 (including airmail postage).

This new magazine from the UK is slick, straightforward, and highly readable. Its publisher says that *Practical Computing* is written for hobbyists, educators, and small business users. "We review equipment, suppliers, software, and applications."

Judging by the sample issue (November) we received, that's a pretty fair description. And true to the name, the approach is practical. A comprehensive, three-page review of the TRS-80 leads off the feature section; at the end of the piece, the pluses and minuses noted in the article are summarized in an easy-to-read box.

That issue also includes a buyers' guide to home computers, covering just about every small machine available in the UK; a "teach-yourself-programming" article, which is one in a series of excerpts from Donald Alcock's excellent book, *Illustrating BASIC*; a report on how PETs can be used in the classroom; and an article, including a listing, on programming Mastermind.

The November "Computabits" column is a grab bag of everything from Kim applications to a discussion of structured programming. "A Practical Glossary," which is a regular department, concludes the issue; this particular list runs the terminological gamut from C to D, defining 27 words falling between "core" and "down-time."

The personal computing movement in Britain is several years behind that of the U.S., and *Practical Computing* reflects this stage of development. However, some of the more technically sophisticated U.S. periodicals would do well to imitate *Practical Computing* in matters of style. PC's writing is tight, well organized, clear—and nearly jargon-free. This makes it a publication that can be understood, at least in part, by the computer novice who has just purchased his or her first machine. As an editor, I find this magazine a first-rate product. □

Reviewed by Louise Burton.

Announcements

Hardware

TRS-80 Printer. Radio Shack has introduced a printer for the TRS-80 system. The new TRS-80 Quick Printer II produces hard-copy output on 2-3/8" wide aluminum-coated paper, in both upper- and lowercase characters. It also prints double-size and double-spaced characters to allow for special effects such as printing headings. Automatic "wrap-around" prevents data loss when the text exceeds the maximum line length, according to Radio Shack. The printer is software selectable for 16 or 32 characters per line and produces 120 lines per minute, 64 characters per second.

Although designed for use with Level II TRS-80 systems, the printer is said to be usable with other computers, too. The Radio Shack TRS-80 Quick Printer II is priced at \$219. It's available from Radio Shack Computer Centers and participating Radio Shack stores and dealers nationwide.

TRS-80 Data Enhancer. A data enhancer that the manufacturer claims "eliminates 99% of all cassette loading problems" on the TRS-80 has been introduced by Microsette Co. of Sunnyvale, CA. Designed for either Level I or II, the data enhancer cleans up and reconstitutes poor quality cassette signals so that cassettes will load reliably with a volume setting of 4 to 10 on the recorder. Since the enhancer requires no modification of the computer or recorder, the Radio Shack warranty is not violated. Data Enhancer, model DE-80, is available for \$45 prepaid (check, money order, VISA, Master Charge) from Microsette Co., 777 Palomar Ave., Sunnyvale, CA 94086.

PET Expander. Commodore PET's memory capacity can be expanded with PEDISK, a high-

speed floppy disk and S100 expansion chassis all in one. According to the manufacturer, CGRS Microtech, the S100 expansion will hold all the extra I/O and memory a PET user could want: printer, telephone interface, modem, and even voice I/O cards. The floppy disk is available with up to 3 minifloppy disk drives (total capacity, 80 KB) or up to 4 full-size disk drives (total capacity, 1 MB). Systems start at \$799.95. For complete details, write: CGRS Microtech, P.O. Box 368, Southampton, PA 18966. (215) 757-0284.

Apple Pen. A light pen for the Apple II computer is now available from Programma International in Los Angeles. This low-cost, simple-to-install light pen has a number of applications, such as bar graphs, charts, and games. It comes with three cassette programs which demonstrate its uses as well as providing aid in developing BASIC programs to drive the pen. The entire package—light pen, software, and operating manual—is priced at \$34.95. For further information, contact Programma International, Inc., 3400 Wilshire Blvd., Los Angeles, CA 90010. (213) 384-0579.

PET RAM Adapter. This 2114 RAM adapter for the PET computer makes it possible to replace two 6550 RAMS with 2114-type RAMS without the addition of a decoder—and at less than half the usual cost. Assembled RAM Adapter (less 2114) is \$6 (plus 20¢ postage); RAM adapter PC board (drilled, instructions), \$2; decoder PC board (drilled, instructions), \$2; instructions only (schematic of both PC boards), \$1. California residents add 6% sales tax. Enclose self-addressed, stamped envelope with all inquiries. For further information, write: Don Henderson, P.O. Box 664, Westminster, CA 92683.

Software

Apple Software Directory. More than 700 programs for the Apple computer are listed in this directory from WIDL Video of Chicago. Listings include description, memory requirements, price, format, and source. Volume 1 of the directory covers business and utility programs; Volume 2, games and entertainment programs. Each costs \$4.95 and is available at many Apple dealers or postpaid from WIDL Video, 5325 N. Lincoln, Chicago, IL 60625. (312) 271-4629.

People's Software for TRS-80. There's good news for TRS-80 users interested in top-notch programs at bargain prices. For \$7.50 you can purchase 77 public domain programs on one cassette tape. This tape, People's Software, is a product of Computer Information Exchange (CIE), a nonprofit organization dedicated to the inexpensive sharing of programs (along the lines of DECUS, an organization for DEC users). CIE emphasizes that People's Software was not brought out to compete with commercial software. In fact, CIE encourages TRS-80 users to market programs, offering free space in its *TRS-80 Bulletin*.

"Just the same, personal computing can be made much more rewarding for everyone if users will share the fruits of their labor—and if a way can be found to distribute them inexpensively, in machine readable form," says CIE. The People's Software tape is available from CIE, Box 158, San Luis Rey, CA 92068. \$7.50 plus 50¢ postage. California residents must add 45¢ sales tax.

PET Programs. PET Software Series One is a collection of 25 programs for the PET computer: 16 games and 9 general programs. All of the programs will run on the 8K PET or the new full-keyboard 16K or 32K models. Series One costs \$24.95 and is available at computer stores or direct from ADP Systems, 95 West 100 South, Logan, Utah 84321. (801) 752-2770.

What's in my Apple? Computer Hardware of San Francisco has introduced a new version of its self-indexing query system, WHATSIT ("Wow! How'd All That Stuff Get In There?"), for the Apple II computer. There are already WHATSITs for the North Star and CP/M systems. WHATSIT answers typed-in questions by referring to disk data

that it automatically stores and revises. It has a typical response time of 2 to 10 seconds. Applications include indexing investment portfolios, music or hobby collections, customer lists, household, or professional files. WHATSIT squeezes at least 2,000 entries onto an Apple disk and cross-indexes them automatically. For more information, contact Computer Hardware, P.O. Box 14694, San Francisco, CA 94114. (415) 647-8518.

FORTH for Micros. The FORTH programming language is now available for use on several micro systems, including Apple II, PET, SWTPC, Sphere, and the TRS-80. According to Programma International, which did these micro versions, FORTH requires only a fraction of the memory capacity needed by other languages (just 6K) and can be placed in ROM if desired. It also runs faster (e.g., 4 to 15 times faster than BASIC) and takes about half as much time in software development.

The basic element of FORTH is a word, comparable to a subroutine, which is drawn from words already defined in the FORTH System's dictionary of 200 words. During programming, new words drawn from the vocabulary can be user-defined. FORTH object code is supplied on cassette with preliminary user's manual for \$35 plus postage. For complete information, contact Programma International, 3400 Wilshire Blvd., Los Angeles, CA 90010. (213) 384-0579.

Creative Computing Tapes. Creative Computing Software now offers a comprehensive line of programs on 21 tape cassettes and two 8-inch floppy disks. The three areas covered are: 1) games and recreation, 2) education and self-learning, and 3) self-appraisal. Tapes are available for the Apple II, PET, Exidy Sorcerer, Ohio Scientific Challenger 1P, and the TRS-80. Floppy disks run on any CP/M operating sys-

tem. Each tape has 4 to 7 programs; each disk contains 51 programs. Most of the tapes cost \$7.95. For a complete list, write: Creative Computing Software, P.O. Box 789-M, Morristown, NJ 07960. (201) 540-0445.

Apple Fun. New programs for the Apple II from Softape include: Bomber!—a HIRES graphics game with fast, detailed animation (\$9.95); Electronic Index-Card File—uses Apple disk for storing and retrieving information such as telephone numbers, recipes, etc. (\$19.95); Appletalker—with 16K of memory or more, you can give your Apple the power of speech (\$15.95); Music Kaleidoscope—uses input from your stereo to create a color light show (\$9.95); Talking Calculator—transforms the Apple II into a talking 10-digit calculator (\$12.95); Apple-Lis'ner—make your own programs with voice recognition (\$19.95). All available from Softape, 10756 Vanowen, North Hollywood, CA 91605.

Text Editor. There's a new text editor available for the TRS-80 Level II (16K) and Sphere 6800 systems. Called PIE (Programma Improved Editor), it is a two-dimensional cursor-based editor with more than 25 commands. The program generates cassette tapes compatible with the TRS-80 Editor/Assembler. The available commands, which permit the cursor to be moved anywhere on the screen, include moving forward or backward a full page, searching for a string, inserting, deleting, backspacing, setting tabs, and page scrolling. Commands are simple to implement, each consisting of a single character depressed simul-

taneously with the SHIFT key. Any command can be preceded by a numeric or string argument.

PIE will soon be available for PET and Apple II computers as well. PIE is sold on cassette and diskette for the TRS-80 and Sphere. Cassette version is \$19.95. For more information, contact: Programma International, Inc., 3400 Wilshire Blvd. Los Angeles 90010. (213) 384-0579.

TRS-80 Disk Expansion. With Patch Pak #1™ from Percom Data Company, the TRS-80 disk operating system can be upgraded to 40- and 77-track mini-disk drives. (TRSDOS is designed for 35-track drives.) Percom says that Patch Pak #1 also improves the TRSDOS by eliminating most of the "silent deaths" of a disk drive motor and solves the problem of interference from the TRS-80 "heartbeat" pulse. Two disk drives are required to apply Patch Pak #1, the TRSDOS system disk being inserted in one drive and the Patch Pak mini-disk in the other.

Patch Pak #1 mini-disk is \$19.95. Percom's toll-free ordering number is: 1-800-527-1592. Pay by check, money order, VISA, or Master Charge. Texas residents must add 5% sales tax. For more information, contact Percom Data Company, 211 N. Kirby, Garland, TX 75042. (214) 272-3421.

North Carolina Hobbyists. All amateur and hobby computer users are invited to attend meetings of the Triangle Amateur Computer Club (TACC) in Research Triangle Park, North Carolina. TACC meets the last Sunday of each month at 2 p.m. in the Dreyfus Auditorium at Research Triangle Institute. For more information, contact TACC, P.O. Box 17523, Raleigh, NC 27609.

Alabamans for TRS-80. The Central Alabama TRS-80 Computer Society "is up and running," writes coordinator Walter Bray. The group meets the third Tuesday of each month at various locations in Montgomery. For more information, contact Bray at 2073 Rexford Road, Montgomery, AL 36116.

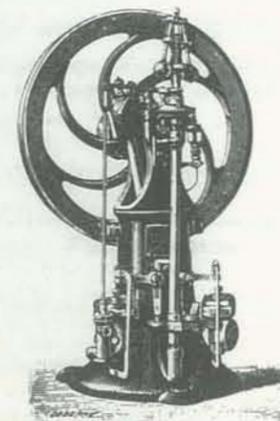
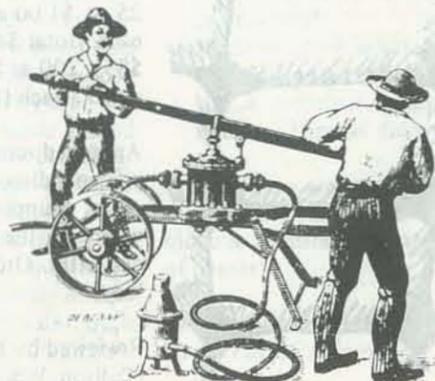
Users' Groups

East Bay Apples. A new computer users' group, ABACUS (Apple Bay Area Computer Users Society) meets the second Monday of each month at the Hayward Byte Shop, 1122 B Street, Hayward CA. Membership is \$12 a year and includes a monthly newsletter. For more information, contact Ed Avelar, president, at (415) 583-2431; or David Wilkerson, secretary, at (415) 482-4175.

Massachusetts Sorcerers. Computer Mart of Massachusetts is setting up a users' group for Exidy Sorcerer owners in the area. For more information, contact Bruce McGlothlin at Computer Mart, 1395 Main Street, Waltham, MA 02154. (617) 899-4540.

North Carolina Hobbyists. All amateur and hobby computer users are invited to attend meetings of the Triangle Amateur Computer Club (TACC) in Research Triangle Park, North Carolina. TACC meets the last Sunday of each month at 2 p.m. in the Dreyfus Auditorium at Research Triangle Institute. For more information, contact TACC, P.O. Box 17523, Raleigh, NC 27609.

Alabamans for TRS-80. The Central Alabama TRS-80 Computer Society "is up and running," writes coordinator Walter Bray. The group meets the third Tuesday of each month at various locations in Montgomery. For more information, contact Bray at 2073 Rexford Road, Montgomery, AL 36116.



Conferences

Holistic Education. The Mandala Society and the National Center for the Exploration of Human Potential are sponsoring a conference in San Diego called: "Mind: Evolution or Revolution? The Emergence of Holistic Education." The weekend symposium is at Town and Country Convention Center, July 6-8; workshops at UC San Diego, July 9-12. Speakers include many nationally known therapists and educators. RC Editor Bob Albrecht will lead a workshop on fantasy gaming. For more information, contact the Mandala Society, P.O. Box 1233, Del Mar, CA 92014. (714) 481-7751.

SIGPC '79. The first annual conference on Research and Development in Personal Computing will be held August 8-10, 1979, in Chicago at the Hyatt Regency O'Hare. The conference is sponsored by the Association for Computing Machinery (ACM) and its Special Interest Group on Personal Computing (SIGPC).

SIGPC '79 will be held during Chicago Computer Visualization Week (August 6-10, 1979) along with the IEEE Pattern Recognition and Image Processing Conference (PRIP-79) and the ACM/SIGGRAPH Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '79). A large trade show of personal computer and graphics equipment is planned to accompany papers, panels, user group meetings, workshops, and person-to-person poster booths. For more information, contact Maxine Brown, SIGGRAPH '79 Exposition, Hewlett-Packard, 19400 Homestead Road, Cupertino, CA 95014.

Northeast Computer Show. September 28-30. Hynes Auditorium, Prudential Center, Boston. For more information, see March-April RC or contact Northeast Expositions, Box 678, Brookline Village, MA 02147. (617) 522-4467.

NYSAE DS. The New York State Association for Educational Data Systems will hold its annual conference at the Granit Hotel in Kerhonksen, N.Y., Oct. 21-23. The conference theme is "Instructional Computing - Hardware/Software/Courseware." For more information, contact Mary Heagney, 9201 Shore Road, Brooklyn, N.Y. 11209. (212) 596-5850.

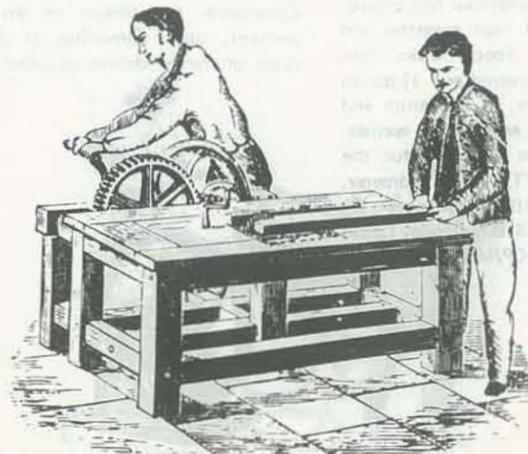
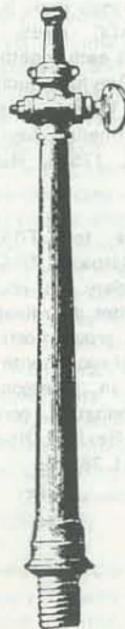
Other

Games Evaluation. Want to put your own program to the ultimate test? Send it to the Marin Computer Center, where it will be reviewed by a "group of hardcore, high school computer game addicts." Just mark the cassette "for review" and mail to the Marin Computer Center, Oakview School, 70 Skyview Terrace, Room 301, San Rafael, CA 94903. The critics' ratings will be tabulated and the results published in *Computer Cassettes*, edited by Robert Elliott Purser, P.O. Box 466, El Dorado, CA 95623. For more information, write Purser or call David Fox at the Marin Computer Center: (415) 472-2650.

TRS-80 Survey. A lowercase character set is the most commonly desired change in the TRS-80, according to a survey of users conducted by the *TRS-80 Bulletin* (published by the Computer Information Exchange). Of the first 162 surveys returned to the newsletter, exactly half requested lowercase letters, one-quarter wanted denser graphics, and about one-eighth wanted faster cassettes, numeric pads, and debounced keyboards. Less than 10 percent wanted color graphics. Surveys went to the newsletter's 4,500 paid subscribers. The editors made no suggestions and created no categories. Replies were totally up to those taking part in the survey.

Leaderless Cassettes. TARZAC/Computer Services is now providing its C-12 leaderless computer cassettes nationwide. The cassette is a true leaderless, 5-screw design, using 3M brand LN recording tape. The problem with leadered cassettes is that leaders don't record, and leaders normally run 18 to 24 inches. C-12 leaderless cassettes are \$1.95 each (in hard storage box, with labels), plus \$1 shipping charge per order. Write: TARZAC/Computer Services, Box 10203, Norfolk, VA 23513.

Teachers Wanted. The Department of Defense Dependents Schools, serving the children of U.S. military and civilian support personnel throughout the world, are looking for computer science teachers and electronic training instructors. Thirty-eight computer systems are available to the instructional program, with BASIC being the most commonly used language. Requirements: a minimum of nine semester hours in computer science plus sufficient course work to qualify as a teacher in another subject. For full information, call Mr. Francisco or Ms. Wycoff at (202) 325-0690, or write for an employment application to: Department of Defense, Office of Dependents Schools, 2461 Eisenhower Avenue, Alexandria, VA 22331. □



Announcing-

PEOPLE'S SOFTWARE:

Tape 1, includes following programs: mortgage calculation-payments, Dow-Jones Industrial forecast, cash flow, inventory and change, California state income tax, journal/ledger (8K), loan amortization, perpetual calendar, bio-rhythm, payroll, diet, speed reading, rock, scissors, paper, seek, Star Trek III (6K), Red Baron, mini-trek, strategy, pilot, battleship, "On a Snowy Evening", math problems, queen, Star Trek I, number guessing, wheel of fortune, World War II bomber.

Plus Level II tapes have the following additional programs: Speed reading, touch typing, sales receipt tally, decision maker, mail addressing, straight-line & double declining depreciation, revolving charge account, mastermind, tic-tac-toe, grand prix, bingo, state capitals, etch sketch, hangman.

TAPE 2:

Fully documented in "Some Common Basic Programs" by Lon Poole & Mary Borchers (Osborne & Associates, 830 Bancroft way, Berkeley CA 94710—\$7.50 plus 50¢ for U.P.S. delivery, else 4th Class Mail), investment, future value regular deposits; regular withdrawals, initial, minimum (for withdrawals), nominal interest, effective & earned-interest; depreciation rate, amount depreciation; salvage value; discount comil paper; loan principal, regular & last payment, remaining balance, term-loan; mortgage amortization; greatest common denom.; integer prime factors; polygon area; triangle parts; analysis, operations two vectors; radian/deg., deg./radian conversion; coordinate; coordinate, polar equation, functions plot; linear, curvilinear interpolation; Simpson's & trapezoidal rules, Gaussian quadrature integration; derivative.

Side 2:
Quadratic equation, polynomial (Newton) & half-interval-search roots; trig polynomial; simultaneous equations; linear programming; matrix addition, subtraction scalar multiplication, multiplication, inversion; permutations & combinations; Mann-Whitney U test; mean, variance, standard deviation; geometric mean & deviation; binomial, Poisson, normal, Chi-square distribution; Chi-sq., student's T-distribution test; F-distribution; linear correlation coefficient; linear, multiple-linear, Nth order, geometric, exponential regression; system reliability; future projections; Federal withholding taxes; tax depreciation schedule; checkwriter; recipe cost; map check; day of week; days between two dates; Anglo to metric; alphabetize.

DEALER INQUIRIES INVITED

PEOPLE'S SOFTWARE: 77-program tape \$7.50

Now TRS-80 owners don't have to shell out a bundle of money or work hard to get a good assortment of programs for the world's most popular computer.

People's Software gives you up to 77 public-domain programs on one cassette tape, just \$7.50 plus 50¢ postage and handling (CA residents add 45¢ tax—FOREIGN orders must be paid in U.S. funds; postage is \$1 per tape, via air).

Nonprofit Computer Information Exchange has been concerned about how public domain software can be easily made available to the public.

Anyone with a library of computer magazines can keyboard-in a wealth of software, at no more cost than the user's time and frustration. CIE's experience has been about 150 hours for the 77-program Tape 2. Assuming that \$1 of the \$7.50 selling price of the software is medium cost (the tape), a person doing the job himself will be saving 4.33 cents per programming hour. Tape 1, even though it contains fewer programs, half of which came from the San Diego TUG (user group), required many more hours preparation.

People's software was not brought out to compete with commercial software. CIE encourages TRS-80 users to market programs. The nonprofit organization gives free space in TRS-80 Bulletin, as well as offering low-cost Bulletin ads. Good commercial software is the foundation of our TRS-80 computing, starting with TRS-80's Micro Soft Level II Basic ROMs.

With introduction of People's Software, personal computing can be made much more rewarding for everyone, since users can share the fruits of their labor—and this can be inexpensively distributed, in machine-readable form (no \$0.0433-per-hour keyboarding!).

Digital Equipment Corp (DEC) computer users, by joining DECUS, have been able to share programs inexpensively. Now Radio Shack users have People's Software.

computer information exchange, inc.

box 158

san luis rey ca 92068

() TRS-80 COMPUTING, \$15/12 issues; \$18us Canada, Mexico; other foreigners \$28us, via air.

() TRS-80 BULLETIN, free (automatic with above), US only.

PEOPLE'S SOFTWARE: \$7.50us plus 50¢ postage (Foreign: \$1us, except Canada & Mexico [50¢US]); CA buyers add 45¢ tax

() Tape 1, Level I (), Level II ()

() Tape 2, "Common Basic Programs"

Make checks payable COMPUTER INFORMATION EXCHANGE

Charge my VISA (), MasterCard () # _____

_____(signed), expires _____

name _____

street address _____

city, state, zip _____

