

93065JOHNSB09

D

2239

BYRON JOHNSON
356 LAGUNA TERR
SIMI VALLEY, CA 93065



Dr. Dobb's Journal
of Computer Calisthenics & Orthodontia



FREE SOFTWARE

COMPLETE SYSTEMS & APPLICATIONS SOFTWARE

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

INDEPENDENT CONSUMER EVALUATIONS

PRODUCT REVIEWS & CONSUMER COMMENTS

Dr. Dobb's Journal publishes independent evaluations—good or bad—of products being marketed to hobbyists. It is a subscriber-supported journal. *Dr. Dobb's* carries no paid advertising; it is responsible *only* to its readers. It regularly publishes joyful praise and raging complaints about vendors' products and services.

REVIEWS

"A publication that is a must for everyone in the hobbyist world of computers. Don't miss it."

'Newsletter'
The Digital Group

"THE software source for microcomputers. Highly recommended."

'The Data Bus'
Philadelphia Area Computer Society

"It looks as if it's going to be THE forum of public domain hobbyist software development. Rating — ☆☆☆☆"

'TRACE'
Toronto Region Association of Computer Enthusiasts

"The best source for Tiny BASIC and other good things. Should be on your shelf."

'The Computer Hobbyist'
North Texas (Dallas) Newsletter

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 \$12 (\$15 after Oct. 1, 1978) to *Dr. Dobb's Journal*, Dept 5E, 1263 El Camino Real, Box E, Menlo Park, CA 94025 or send in the postage-free card at the center of this magazine.

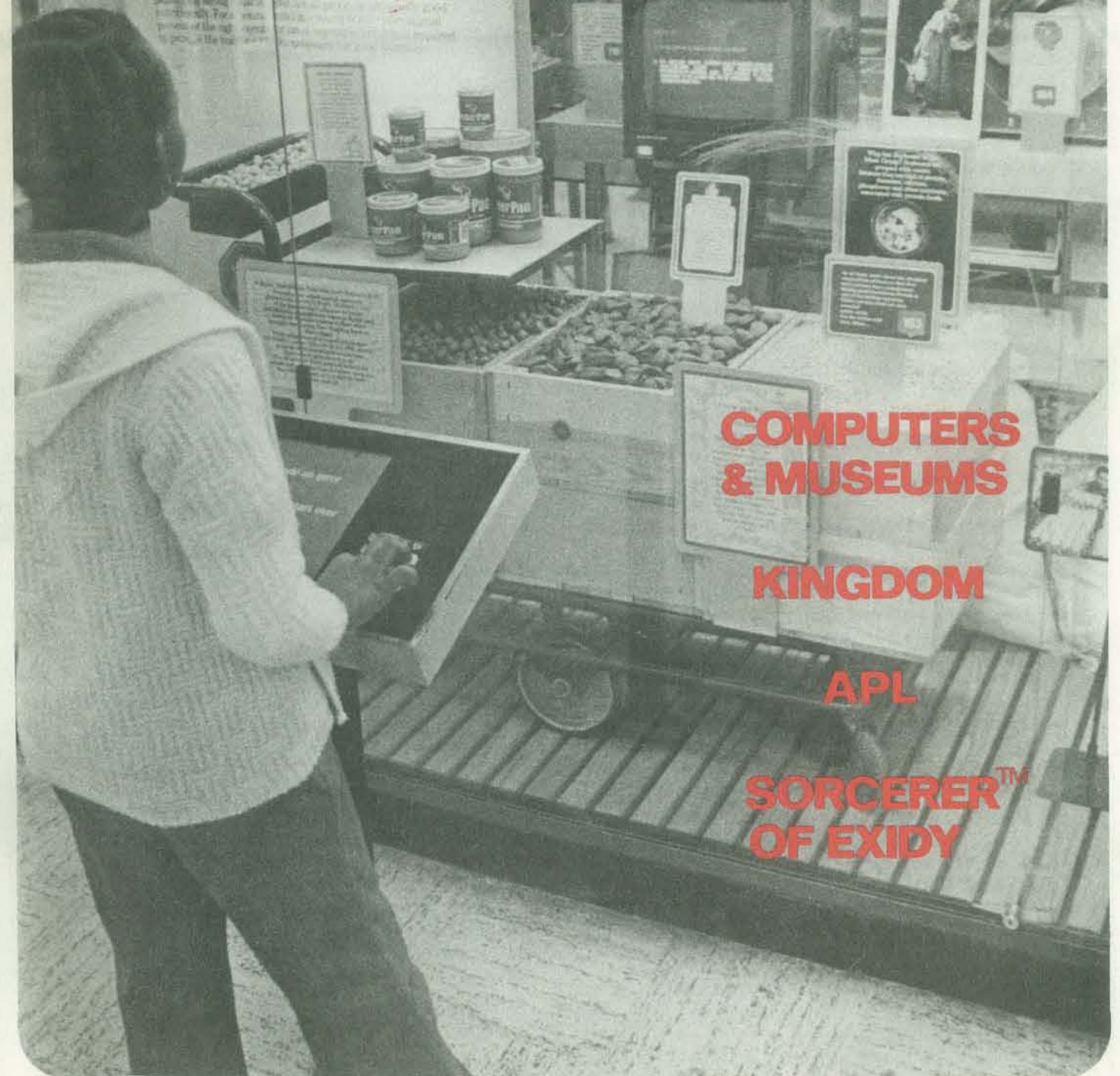
people's computers

\$1.50

VOLUME 7 NO 2

SEPTEMBER - OCTOBER 1978

Soon-to-be RECREATIONAL COMPUTING



SUBMITTING ITEMS FOR PUBLICATION

LABEL everything please, your name, address and the *date*

TYPE text if at all possible, double-spaced, on 8½ x 11 inch white paper.

DRAWINGS should be as clear and neat as possible in black ink on white paper.

LISTINGS are hard to reproduce clearly, so please note:

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

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

CIRCULATION NOTE: To decipher the expiration date of your subscription, look at the top right hand corner of your address label. The last two digits refer to a code/issue number that is your expiration date. Hence, read 35 as 9/78, 36 as 11/78, 37 as 1/79, 38 as 3/79, 39 as 5/79, 40 as 7/79, and so on.

Cover photo: A visitor using the computerized nutrition exhibit at the Museum of Science & Industry in Chicago. See article on pp 30-33.

People's Computers 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 © 1978 by People's Computer Company, Menlo Park, California

SUBSCRIPTIONS

U. S. Subscriptions

- \$8/1 yr. (6 issues)*
- Retaining subscription @ \$25 (\$17 tax deductible)
- Sustaining subscription @ \$100+ (\$92+ tax deductible)

*After Nov. 1, 1978 \$10/yr.

Foreign Surface Mail

- add \$4/yr. for Canada
- add \$5/yr. elsewhere

Foreign Airmail

- add \$8/yr. for Canada
- add \$11/yr. for Europe
- add \$14/yr. elsewhere

Payment must be in U.S. dollars drawn on a U.S. bank.

These back issues are available at \$1.50 each:

- Vol 5, No 6
- Vol 6, Nos 1, 2, 3, 4, 5
- Vol 7, No 1

Foreign Distributors of *People's Computers*

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 St
Sydney NSW 2000

ASCII Publishing
305 HI TORIO
5-6-7 Minami Aoyama
Minato-Ku, Tokyo 107
JAPAN

Eastern Canada
Liz Jänik
RS-232
186 Queen St W
Suite 232
Toronto ON M5V 1Z1

Western Canada
Bill Blake
PACOM
4509 Rupert St
Vancouver BC V5R 2J4

Integrated Computer
Greenhills PO Box 483
San Juan, Metro Manila
PHILIPPINES 3113

people's computers

VOL 7 NO 2
SEPT-OCT 1978

SOON-TO-BE RECREATIONAL COMPUTING

STAFF

EDITOR
Bob Kahn
ART DIRECTOR
Meredith Ittner
PRODUCTION
Sara Werry
ARTISTS
Matthew Heiler
Ann Miya
Judith Wasserman
TYPISTS
Phyllis Adams
Renny Wiggins
CIRCULATION
Michael Madaj
BULK SALES
Christine Botelho
DRAGON EMERITUS
Bob Albrecht
SPOT EDITOR
Phyllis Cole

RETAINING SUBSCRIBERS

David R. Dick
Mark Elgin
John B. Fried
W. A. Kelley
Frank Otsuka
Shelter Institute
Brett Wilson

SUSTAINING SUBSCRIBERS

BYTE Publications
Paul, Lori and Tom Calhoun
Dick Heiser, The Computer Store

And a special thanks to all the folks at People's Computer Co.: Curtis Abbott, Chuck Bradley, Claire Connor, Delia Daniels, Cary Helvey, Willard Holden, Cynthia Kosina, JoAnn Loeffler, Ann Merchberger, Kathleen Shaw, John Snell, John Strawn, Eryk Vershen, Tom Williams, Denise Winn.

SPECIAL FEATURES

- 8 THE INVASION OF KINGDOM by Lee Schneider and Todd Voros
Teaching a computer to play its own game . . . and win!
- 20 COMPUTERS AND SCIENCE MUSEUMS — PART II by Bob Kahn
Hands-on public access to interactive computing
- 30 COMPUTERIZING MUSEUM EXHIBITS by Bruce Burdick
Facilitating a dialogue between the museum and the visitor

ARTICLES

- 6 TRS-80 LEVEL II: A GROWN-UP FIELD EVALUATION
by Thomas Dwyer and Margot Critchfield
A review of Radio Shack's Level II computer
with additional comments by The Dragon and Eryk Vershen
- 16 SNOOPING WITH YOUR PET by Mark Zimmermann
Making full use of machine language capabilities of 6502 systems
- 36 APL AS A TINY LANGUAGE by Mokurai Cherlin
Sizing up APL with Sam Hills's specs for a model tiny language
- 50 THE SORCER™ OF EXIDY by Bob Kahn
A first look at Exidy's new microcomputer system

MICRO SYSTEM APPLICATIONS

- 34 DECIMALS IN TINY BASIC by Milan Chepko
Simulating decimal arithmetic in Denver Tiny BASIC
- 41 APPLE-MATH by John Gaines
A math drill program for the Apple II
- 44 SPOT: The Society of PET Owners and Trainers by Phyllis Cole, Ed.
A compendium of commentary and programs on the PET

DEPARTMENTS

- 4 EDITOR'S NOTES AND LETTERS
- 28 FORTRAN MAN
- 27 REVIEWS
- 49 DRAGON SMOKE
- 52 ANNOUNCEMENTS



LETTERS

EDITOR'S NOTES

I found a typewriter this issue!

We have reached a turning point for People's Computers. Note our new cover logo. Indeed, we are planning to change our name to Recreational Computing to emphasize the fact that in future issues we intend to focus on computing as a recreational and educational pastime. This was a major role that PCC Newspaper foresightfully fulfilled for schools; now, retaining our new 'magazine look,' we intend to address ourselves to the main use of computers in the home—games, programs and diversions for entertainment and home education. So, in coming issues, look for more programs and articles on fantasy/adventure games and simulations, more computing problems for you to solve and diversions for you to puzzle over, more on graphics and more on packaged microsystems—in black & white... and color!

This issue is a few pages thinner than usual, although it's unusually thick in 'meaty' articles. We are currently striving to get on an earlier production schedule in order to counteract the inevitable delays the magazine suffers between the time it leaves the printer and the time it actually appears on the newsstands or in your mailbox. To get on this new schedule required some fast editing and a little less copy (although, I must admit, a non-multiple of 16 pages has given PC an interesting new look in terms of color pages). Not to worry, however; future issues will be normal size.

Finally, I must pass on this message which I received in a fortune cookie (at our favorite local Mandarin Chinese restaurant) yesterday:

You may have to be patient now—
think, listen and heed signs.

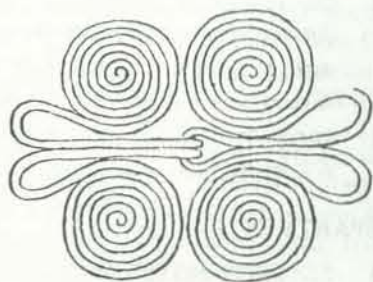
Bob Kahn

I just received your July-August publication and was paging through it when I saw the article on 'A Modern Day Medicine Show.' I graduated with you, Fred and Randy from GW in 1964, so reading the article was quite a pleasure as I have fond memories of those days.

I moved to California in 1970 and have been employed by Pertec Computer Corporation since that time. I am currently working on the MITS product line and its international distribution.

Jan J Cunningham
Irvine, CA

Jan—At least 75% of my mail consists of press releases and new product announcements, so I was absolutely delighted to open what I expected to be a press release from Pertec Computer Corporation to find your letter. I had been wondering what the chances were that at least one of the 1000 members of our high school graduating class are among the 10,000 readers of People's Computers... obviously, very good. What an outstanding way to have a class reunion. —BK



I wish to congratulate you on the outstanding July-August 1978 issue. You packed substance and enjoyment into each article.

Gene Beley
Arcadia, CA

I wanted to write and express appreciation for your latest SPOT program (the 'Pong' one) and PC in general. British PET clubs are non-existent, and as a really new newcomer to computing, with no contacts, PC is a lifeline! More programs, and keep growing!

Duncan Langford
Cochester, Essex, England

I am pleased to see that your magazine is carrying material which is of interest to those of us without PETs. As a TRS-80 owner, I hope to see more articles along the lines of Clyde Farrell's 'TRS-80: Converting to Level II BASIC' (PC, Jul-Aug '78).

While the TRS-80 is not the 'quick and cheap machine' described by Tom Williams (PC, Mar-Apr '78), neither is it perfect. I have an early version of Radio Shack's Level II BASIC and, while it is generally an excellent extended BASIC, it does have a few 'bugs.' I was having trouble READING DATA statements until I received the following 'fix' from Hugh Mathias, Radio Shack's Customer Service Manager. With the insertion of a line at the beginning of a program (Line #POKE 16553, 255), Level II READS DATA without a hitch. Others with Level II should appreciate this 'fix.' Incidentally, it would seem logical that other versions of Microsoft BASIC have similar 'bugs.'

I feel that the TRS-80 system could fill a need for inexpensive hardware to handle limited applications for smaller governmental units and small businesses. I am now awaiting delivery of two mini-disk systems and a line printer so that I can test my theory, and I'll keep you advised of any successes (and/or failures). I would be most interested in hearing from others regarding their experiences in implementing microcomputer systems for governmental or commercial applications.

William S Pitt
Pacific Grove, CA

I must add my voice to those who have written in support of TRS-80. I have had mine since November and have had Level II since May, it has never failed to please.

At the same time I must express consternation over what seems to be developing between users of the PET and TRS-80, namely some sort of limited warfare. As Clyde Farrell has stated before, we are lucky that two companies have now made personal computing within the financial grasp of so many. Engaging in protracted infantile debates (see for example the letter by Dave Caulkins in PC July-August) is a waste of time and paper.

It is obvious that both machines have their faults as well as their attributes; why, for example, compare a machine with 4K ROM to one with 14K ROM? It doesn't make any sense.

As PC points out, and I guess it is true, the space dedicated to a particular unit is a function of the amount of correspondence received. They say they are willing to give 'equal time' to the TRS-80 and so far as I can see they are attempting to do so. So if you TRS-80 owners out there want more coverage, send in more programs, comments, etc.

I continue to find PC a refreshing publication, keep up the good work. Any microcomputer owner should be happy to have access to a magazine such as yours.

Arthur B Busbey, III
Chicago, IL

I'm really impressed with Milan Chepko's *Tiny Blackjack* (PC, May-June, 1978) but believe that the card dealing routine suggested isn't fair, in that if (for example) a deck with 4 Aces and 1 King is left, the King is as likely to be dealt as the Aces. If, as in actual play, there are more cards left, the bias is less severe, but still exists.

A possible fix is to generate a random number between 1 and N and to take the card with that position in the (now depleted) deck to be dealt. My attempt to do this is shown below. I don't know exactly what Milan's version of BASIC does for RND(0) or for division, so the

suggested program may be flawed—I would appreciate it if you would check it out and let me know (I have only a PET).

Except for the card generator, I like Milan's BJ Program a lot! I worked for awhile developing BJ on my own a few months ago, but never got done with a version that allowed doubling down, pair splitting, insurance, etc. My first card shuffler subroutine had severe problems—that's why I got nervous about this one and started analyzing it.

FAIR DEALER FOR TINY BJ (CHEPKO)
Assuming RND(0) produces an integer between 0 and Y (is Y=32767 or 65535?) and that division truncates and does not round off (i.e. 3/2=1 and 7/8=0) then the following replacement for lines 160-165 in the Chepko program deals fairly:

```
160=RND(0)/(32767/N)
161 IF X>(N-1) GOTO 160
162 I=0:J=0
163 I=I+1:J=J+S(I):
164 X=X/4:S(I)=S(I)-1:N=N-1
165 IF X=1 X=15
166 RET
```

Mark Zimmermann
Sierra Madre, CA

Notes to Zimmermann in Response:

I think Mr Zimmermann raises a valid point in his letter—my card-drawing routine (lines 160-165) does not take into account the changing frequency of the remaining cards as the hands are dealt, and I can see how this could affect the probability for any card drawn next to be a specific type.

While I agree that my original routine may upset a statistician, I doubt that there would be any serious problem in routine play, although I would be interested in seeing a good analysis of the situation. It would seem to me that if the random number generator functions correctly, the order of the cards would be no less random than the order of cards in a well-shuffled deck, and that is the basis of the card game in the first place.

Milan D Chepko, MD
Thief River Falls, MN

Denver Tiny BASIC generates a number between 0 and 32767 for each call to RND(0). The only kind of division in DTB is integer or truncated division.

You are indeed right that the card dealing routine in Chepko's *Tiny Blackjack* is not fair. Your fix should work.

I don't recall noticing the problem when I checked the program; however, just to prove that I'm a programmer...and to give you a choice, I've included another possible fix.

```
160 X=RND(0)/630+4
161 IF X>55 GOTO 160
162 IF S(X/4)<(X-(X/4)*4)
163 GOTO 160
164 X=X/4:S(X)=S(X)-1:N=N-1
165 IF X=1 X=15
166 RET
```

Eryk Vershen
PCC



Corrections to last issue
(Volume 7, Number 1)

- 1) In Isaac Asimov's article there should have been a copyright notice on p. 16 which read:
©Copyright 1969 by Communications/Research/Machines, Inc. Reprinted with permission of the author.
- 2) In the Farrell article on page 30 the address should have read:
Farrell Enterprises
PO Box 4392
Walnut Creek, CA 94596
- 3) A couple of errors in last issue's SPOT section are corrected in this month's SPOT section.

TRS-80 LEVEL II COMPUTER: A GROWN-UP FIELD EVALUATION

BY THOMAS DWYER AND MARGOT CRITCHFIELD

WITH ADDITIONAL COMMENTS BY DRAGON BOB ALBRECHT AND ERYK VERSHEN

Now that Radio Shack's Level II computer has reached the marketplace, and in response to the letters and articles in our last three issues, the TRS-80 fan mail is starting to roll in. (I knew that there were more of you TRS-80 supporters out there.)

Tom Dwyer is a professor of Computer Science and the father of Project SOLO (now SOLOWORKS) at the University of Pittsburgh. Margot Critchfield is an artist and graduate student as well as the co-director of the SOLOWORKS project. In a sentence, the aim of Project SOLO (SOLOWORKS) was to involve kids in learning to program and use computers as tools for creative problem-solving, particularly in mathematics.

More recently, Tom and Margot have shifted their interest to personal and home computing, and they have published a number of articles on this subject (particularly on using the Compu-color 8051 to produce color graphics) in various popular computing magazines over the past few months. Their book, BASIC and the Personal Computer, which was just released by Addison-Wesley last May, will undoubtedly become a classic reference volume.

I ran into Tom and Margot at NCC in Anaheim a few weeks ago, and we had a long chat about consumer-oriented micro-computer systems. A couple of weeks later I was delighted to open my mail and receive this article on the TRS-80 Level II. Bob Albrecht read over the manuscript and decided to add a few comments of his own, and Eryk Vershen, who has been busily checking out the Level II machine here at PCC, volunteered a few more comments based on his experience. —BK

The Radio Shack TRS-80 computer first appeared in the Fall of 1977. At that time it used a 4K version of BASIC (stored in ROM) which Radio Shack designated as Level I BASIC. Some of the

early reviews of the TRS-80 pointed out that prospective buyers ought to view this BASIC as a negative factor in judging the machine.

In April of 1978 Radio Shack started to ship its Level II computer. The Level II features a greatly expanded version of BASIC (stored in about 12K of ROM). It was written by Microsoft, and it closely resembles the 4.0 BASIC that Microsoft produced for the ALTAIR™ computer. What this amounts to saying is that Level II represents current state-of-art thinking in extended BASIC design, and that the TRS-80 now has the potential of being rated a best buy for many users.

We recently had a chance to evaluate this potential in the context of teaching two short adult courses in Personal Computing at the University of Pittsburgh's Pitt Informal Program. The courses were intense introductions to all the aspects of personal computing: hardware, software, programming, choosing a machine, and the design of applications. Each course was conducted over four successive Saturdays, with optional 'lab' time during the week.

Two kinds of lab computing were used. All students were issued identification numbers on the Pitt Timesharing System where they learned to program in BASIC using a Decwriter™ terminal to 'simulate' their own personal computer. We also tried to do actual classroom programming using timesharing. This proved to have lots of problems. We wanted to show students how to develop programs by trying things on-line. Large, expensive timesharing systems can't permit very much 'think time' however, and having the system log you off just when 'things are starting to jell' proved very frustrating. Also, the BASIC on this large system (a DEC PDP-10) is relatively primitive (the BASIC PLUS language on the smaller DEC PDP-11 RSTS system is far superior).

The solution to these problems came in the form of a Radio Shack Level II machine that was delivered to David Stanton, one of the students in our class. Dave kindly volunteered to make his machine available for course use. The modularity of the system meant that he only needed to bring in the keyboard/computer unit, which we then hooked up to our classroom monitors. This simply involved making up a coax connecting cable, since the TRS-80 (unlike the PET) provides a standard composite video signal on the back panel.

After running the machine through its paces on a number of difficult assignments, both we and the students gave the machine high grades on just about all counts. Here are some of the reasons:

- A really superior BASIC with excellent editing and diagnostic facilities.
- A medium resolution graphics feature that can address any of 128 by 48 points. (The PET graphics addressability is about 1/6 this.) One of our most successful class activities was the on-line development of a dynamic billiard ball program using this graphics capability.
- A 1024 character alphanumeric display with 64 characters on each of 16 lines.
- A composite video signal coming out the back that is able to drive several classroom monitors.
- Software selectable double-sized characters for classroom viewing when needed.
- Portability. (Dave carried to computer/keyboard unit, power supply and cassette player boxed together with a rope slung over his shoulder.)
- Room for very large user programs (we used the 16K RAM version).
- No known bugs and no hardware failures; easy local access to service if needed.
- All of the above for under \$1000.


The main questions one might have about the TRS-80 are connected with expandability. Most of our students

wanted to use computers in applications that would require both disk storage and hard copy. These facilities have been announced by Radio Shack, and their 32K business system in particular will be worth investigating as the answer to this need. We won't speculate on how well the business system performs without a similar 'field' experience. There's no doubt that trying out a system in the context of keeping 40 critical adult students happy with the results is a demanding form of evaluation. But it's one that the TRS-80 Level II passed with flying colors.

SOME DRAGON-THOUGHTS ON THE TRS-80

I agree with Tom and Margot—the TRS-80 is a great computer for teaching people how to program in BASIC. This is especially true if the people are kids, ages 8 to 12. The TRS-80 is my current first choice for teaching kids how to program in BASIC.

Tom and Margot listed several reasons for liking the TRS-80. I concur—and here are some additional reasons, if you are teaching kids how to program.

- The RND function gives integers. For example, suppose you want integers from 1 to 100.
TRS-80: 120 X=RND(100)
Of course, Easy!
Brand X: 120 X=INT(100*RND(1))+1
Try to explain that to a 4th grader.
- Clear the screen.
TRS-80: 100 CLS
Pretty obvious.
PET: 100 PRINT " " 
Very mysterious.
- Cursor control. The TRS-80 with its PRINT@ for text and SET and RESET for graphics is very easy to learn or teach. The PRINT@ can also be used as PRINT@ 64*R+C where R is the row (0 to 15) and C is the column (0 to 63). Neat!
- Double precision arithmetic—up to 16 decimal digits. Great for number patterns and math recreations taught in elementary schools. Kids love to see those big numbers crunched out.

• ON ERROR GO TO. A must for computers used in education. This allows the programmer to be in control, instead of the locked-in-ROM operating system. The TRS-80 has it.

• TRS-80 documentation is outstanding. The Level I "Teach Yourself" style primer is the best I have seen from a personal computer manufacturer. The Level II BASIC Reference Manual is clear, complete and readable. I hope Radio Shack will continue this excellent track record with a Level II Primer.

Enough for now, except for this: Don't misread me, or read more into the above words than I intend. I think the TRS-80 is the best low cost computer (circa September 1978) for teaching people how to program in BASIC. That's all. I didn't claim that the TRS-80 is the best educational computer for all purposes. For some educational applications, I like the PET; for others, the SOL. Next school year, I'll try out the APPLE. In the meantime, what do you think? Send us your thoughts and, especially, your experiences in using various computers with learners of all ages. — The Dragon

A SECOND LOOK AT THE TRS-80

The TRS-80 is definitely 'the' computer system this year. I have now had the opportunity to use both the Level I and Level II machines. Here are a few, more or less objective, comments.

Hardware. The hardware is the same for both models. The only difference lies in the chips inside the keyboard. The standard TRS-80 comes in four pieces:

- A 12" monitor
- A tape recorder/player (Radio Shack's CTR-41)
- A keyboard cum computer
- A power supply.

Setting up is really quite simple. The monitor, recorder and power supply all plug into the keyboard at one end and into wall sockets at the other end. All the pieces are encased in grey and black plastic. The unit as a whole is fairly light weight and any difficulties in carrying result from trying to carry four things at once, not from heaviness. Perhaps the only annoying thing about the hardware is an occasional high pitched

hum coming from the transformer and/or the monitor; however, this hum usually goes away after a few minutes.

Software. Good software is crucial to any computer system. If the hardware doesn't work you can get it fixed or replaced or get your money back. If the documentation is poor, you can often play detective. But if the software is poor you may well be headed for utter frustration. Indeed, it is the software which really distinguishes the Level I and Level II machines.

Level I is a 4K BASIC; that is, it takes 4,096 bytes of ROM. If you have used Palo Alto Tiny BASIC, you will find Level I BASIC extremely familiar—they are practically identical. TRS-80 Level I BASIC allows larger numbers, a few graphics functions, a couple of strings (which can only have values assigned to them, nothing else), and a few additional commands and functions (PRINT @, TAB, DATA, READ, RESTORE, and ON). Nevertheless, Level I is still a rather primitive language, with some definite drawbacks. For example, all input is printed on the screen, even if the machine is not accepting it; furthermore, it is impossible to tell if a program has been correctly saved without destroying the copy in the machine.

Level II BASIC is in a different league. It was written by Microsoft, the same company that designed and implemented BASIC for the ALTAIR™, PET™ and a number of other well-known micro-computers. Level II is a 12K BASIC, and as one would expect with three times the size, it is a *much* better and more sophisticated BASIC than Level I. Variable names can be longer (though only the first two characters are checked and no substring can be a reserved word); a good set of arithmetic functions are provided (including, thank goodness, the exponentiation operator that Level I didn't have); real strings and string functions (though INSTR is left for Level II disk BASIC) are available; there are three numeric data types: integer, real and double-precision real; multiple arrays are allowed; and finally, PEEK, POKE and machine language routines may be used.

The drawbacks of Level I have more than been surpassed. Referring back to the ex-

Continued on page 27.



The Invasion of KINGDOM

BY
LEE SCHNEIDER AND TODD VOROS

OR
HOW TO TEACH
A PERFECTLY
INNOCENT
COMPUTER
TO
TAKE OVER
THE WORLD

In the Sept. 1975 (Vol 4, No 2) issue of PCC, we published an article by Todd Voros and Lee Schneider (creators of FORTRAN MAN) on the history, development and human reaction to their computer simulation game, KINGDOM. Todd and Lee created KINGDOM as an expanded and more sophisticated take-off on the popular program, HAMMURABI developed at Digital Equipment Corporation.

Here, at last, is a long-awaited follow-up article in which Todd and Lee discuss the process by which they designed KINGDOM and subsequently developed programs to simulate various human strategies for playing the game—searching for an optimal strategy.

Since the original article appeared some time ago, we have reprinted the 'official rules' of KINGDOM and the program listing as part of this follow-up article. If you are not familiar with KINGDOM or if you missed the first article, you may wish to peruse the 'official rules' prior to reading this follow-up. —BK

THE DESIGN OF KINGDOM

When we first started to design the game of KINGDOM, we decided to follow the general guidelines of the much older DEC game, HAMMURABI, but to include considerably expanded capability. Three overall criteria governed the design of

the game: 1) the game had to be complex enough to provide an interesting challenge to the experienced player, but not so complex as to be incomprehensible to the novice, 2) the game had to include enough variation in play such that no two games would be exactly alike, and 3) in the overall play of the game, the winning or losing of the game had to rest primarily on the skill of the player, rather than on random chance.

The first criterion was purely an arbitrary selection, since 'complexity' depends on the average player's capability: a game that is difficult for an average fourth-grader may seem relatively simple to a twelfth-grader. We designed our game of KINGDOM so as to be challenging for high-school or beginning college students.

The second and third criteria were somewhat conflicting in nature: variations in the course of play imply the use of random elements; however, the introduction of random elements limits the player's need to use skill. Indeed, too much randomness in each game prevents a player from improving his play from one game to the next, as he can never truly judge the efficiency of his tactics. On the other hand, little or no randomness is dull since each new game looks exactly like the last, and the player quickly loses interest once the 'winning strategy' has been found. Thus, a completely deterministic game must be made so complex that the winning strategy cannot easily be found (chess is

an excellent example of such a game), but this violates the first constraint of keeping the game at a relatively simple level. (Moral: Designing a game isn't always as easy as it looks.) We attempted to solve this dilemma by designing KINGDOM on the basis of 'balanced randomness,' that is, over the course of the typical game, the random occurrences which aid the player and those which oppose him would tend to average out to approximately zero. As a result, the skill of the player (in the long run) determines the winning or losing of the game, while a steady flow of random elements is maintained to keep the game interesting.

Examining KINGDOM, it can be seen that, on the negative side, there are three 'natural disasters' which may strike the kingdom: plague which reduces population, earthquakes which reduce land holdings, and theft which reduces the supply of grain. Each of these set-backs has a 10% chance of occurring in a given year; therefore, during the course of a typical 100 year game, the player can expect a total of 30 'natural disasters' to strike the kingdom. (If you are thinking to yourself, 'Aha, they forgot the Huns!', be patient; we have purposefully ignored them momentarily.) On the positive side, there are only two random occurrences: grain shipments which increase the amount of grain on hand, and border expansions which increase both the land and population of the kingdom. Both of these have a 15% probability of occurrence, resulting in

a total of 30 'miracles' which the player could expect to occur during a given 100 year reign. The overall effect of 'natural disasters' versus 'miracles' is, therefore, zero!

We now turn to the delinquent Huns; these guys, like all other 'bad news' occurrences, have a 10% probability of cropping up in a given year. Their appearance means the loss of all three commodities, population, land, and grain. Therefore, with no other 'good news' occurrences to balance, the average game has a 10% overall bias against the player. This, by the way, was not included in the first versions of KINGDOM; the Huns were added later for a specific purpose (as were other additions). The reasoning was as follows: as the rules were initially set, even the most novice player could make an average of 10% profit during each 'year' of play without much real thought; this meant that a player could win the game without really doing any 'tactical thinking'. By giving the computer a 10% edge, a player is required to do some creative thinking to win the game and not simply 'coast along' for the ride. The amount of destruction wreaked by the Huns was carefully set to a level too large to be ignored by the player, but too small to be of any overall consequence unless the player consistently disregarded their attacks; in other words, the Huns crop up just enough to be a minor annoyance and keep a player on his toes and thinking (which was the whole idea). This design strategy was further enhanced in the final selection of equations governing agriculture and population.

The population of the kingdom is, of course, of vital importance to the player; if all the people die out, the game is lost. However, people in KINGDOM are expensive; too large a population becomes a tremendous burden on the ruler and cuts deeply into profits. The populace can return part of this expense back to the ruler through planting and harvesting of land; however, as most KINGDOM players will soon notice, such farming makes the population almost, but not quite, self-sufficient. This, once again, prods the player into doing some tactical thinking in order to make enough profit each year to maintain the populace.

For those who like numbers, this is how it works. In KINGDOM, each subject

may plant a maximum of 2 acres of land. Each subject requires 10 bushels of grain for food each year, so the 'cost of labor' can be thought of as 5 bushels per acre. In addition, each acre planted requires 3 bushels of grain for seed. Thus, the net cost of planting is 8 bushels per acre. The harvest ranges from 1 to 9 bushels per acre, and is based on a standard, 'bell curve' distribution with the median at 5 bushels per acre. That is, over the course of the game, the average harvest will be 5 bushels per acre. Now, figuring yield minus cost, for every acre planted, the ruler will, on the average, end up 3 bushels in the hole!—On the average, that is, unless the player learns to predict the harvest by observing the distribution of high and low harvest years and 'out-guesses' the weather. Once again, the computer has forced the human being to think in order to get ahead.



We conclude the discussion of the design of KINGDOM with an admission of guilt: there is a 'fix' in the program which is deliberately designed to make things tough for the novice player. As each game begins, a player is given, at random (but within defined limits), a starting allotment of grain, land, and people. It is here, with the people, that the 'fix' occurs: the starting population is almost always a little bit too large to be supported by the resources provided. The novice player is generally tempted to support the population as it is, thereby losing the game as funds slowly run out faster than they can be accumulated. In KINGDOM, by the way, the population tends to increase geometrically (just like real populations do), so that if left unchecked, it will eventually reach a point where mass starvation is unavoidable.

There is, though, a way out of this 'fix'. Just letting the population starve in the beginning is not the answer; a player who

does this, will be wiped out by food riots. Waiting for the plague or the passing Huns to do the job might work, but the odds are against it (border expansions, which occur at a comparable rate, tend to restore the population right back again). The secret is 'controlled starvation'; that is, a player must convert almost all of his grain to land when the price is relatively low, saving just enough to feed the desired number of people. Now, since a food riot results in loss of grain, and since the Royal Grainery is empty, a player may get by with no loss of wealth and also restore his grain by selling land in the next year or so, when the price of land is equal to or greater than it was when the land was bought. Some players, of course, claim that it is 'inhuman' to starve people in pursuit of wealth, while others, (with less conscience and more greed) use the tactic continuously. The former sometimes prefer to use the 'Ruler's Revenge Law' which allows the ruler to eliminate up to two subjects per year from the populace without risking riots, (which may or may not work, depending on how fast the populace is growing).

Overall, the game is designed so as to be somewhat difficult for the beginning user. But, by knowing and playing odds, by intelligent speculation on things to come, and by taking advantage of the known factors which control the game, a win is almost always possible. (Some games, of course, will be unavoidably lost when, by chance, a number of negative occurrences all combine in one 'year' so as to eliminate the entire population in a single stroke). It should be noted that KINGDOM was designed around the use of integer quantities only, since this simplified both the player's input and the internal bookkeeping of the game.

THE SEARCH FOR AN OPTIMAL GAME STRATEGY

We now turn to what we believe is the most interesting development of the entire KINGDOM project—the design and implementation of a program which would enable the computer to play the game by itself, analyze its play, and eventually, be capable of modifying its tactics so as to 'learn' the game as a human player learns it. We embarked on this project, which we called RAMSES, partly because at the time, we had nothing better to do, partly because we had

never attempted to write a heuristic program before ('heuristic programming' is a term used to describe the computerized simulation of human functions, such as learning), and mostly because we hoped to be able to use the computer to find the 'best' strategy for winning the game.

Once KINGDOM had been fully developed, debugged, and released into the unsuspecting world, everybody who played more than two games came running to us claiming that he or she had discovered the 'secret strategy' for assuring a win every game. Others came to us with alternate strategies which, they claimed, would allow the player to win in the shortest possible time (though not necessarily every time). No two people, it seemed, could ever agree on what the 'best' strategy was, or even on the exact nature of the goal in question.

Occasionally, someone would come up with what appeared to be a sure-fire, bona-fide strategy for winning at KINGDOM. Even so, because of the planned variations in the game, it was difficult to evaluate each strategy in terms of effectiveness, and still harder to compare one strategy with another in a meaningful way.

Being Engineering Majors, we knew, of course, that the solution to this problem was to run a large number of games (say a few thousand) using each strategy, thereby obtaining enough random variations to cause an 'averaging out' effect. The results of any one game could not be used as significant proof that a given strategy worked, nor could the results of any ten games; but the results of several hundred games would indeed yield meaningful values, and, the more games played, the better the analysis. Furthermore, it occurred to us that our trusty computer could do all this work, provided that somebody taught it how to play KINGDOM.

TEACHING A COMPUTER TO PLAY KINGDOM

Thus, the first phase of the RAMSES project essentially began as the development of a program which would play KINGDOM as a human player does, following a defined strategy which we would provide. This program, which we named PLAYER, could play large num-

OFFICIAL RULES FOR

Kingdom

The game of KINGDOM is based upon the premise that the player is to act as the ruler of a mythical, medieval country. The ruler must make decisions which maintain the kingdom and advance the player toward the goal.

The goal of the player is to rule the world. This is accomplished by collecting a total wealth equal to, or exceeding, one million units. Wealth is measured in terms of two quantities: acres of land owned, and bushels of grain in storage.

At the start of the game, the player is assigned an initial population, a parcel of land, and a storage house of grain. At the beginning of each year, the player must make decisions which will concern the transaction of land and grain, allot food to the populace, and specify the amount of land to be planted. A yearly report is printed at the beginning of each year, giving the total resources of the kingdom at that time.

If the player makes a decision which exceeds the limits of the total resources at that time, the player is informed of the error and the question is repeated.

The player wins at any time when the total wealth (acres plus bushels) equals or exceeds, one million units. The player loses if the total population reaches a value of zero.

Game constants are of three basic types. The first are absolute constants, which are fixed at the same value throughout the play of the game. The second type are factors which are always present, but change on a fairly random basis each year. The third type is usually referred to as random occurrence, which governs a number of happenings which may or may not occur in a given year, based upon a standard fixed percentage.

All user input is in the form of numeric values. The user may terminate the play at any time by entering a negative value in response to any question.

bers of games using the given strategy and would then allow us to evaluate the effectiveness of that strategy as a means to obtaining a certain goal (which we could also define), with the assurance that occasional random events would not affect the overall evaluation of strategy.

Needless to say, it was necessary for PLAYER to play within the rules of the game, no matter what sort of outlandish strategy it was given to follow. In a totally objective observation of KINGDOM, it could be said that the function of the human player is to provide four integer values as input to the program at the beginning of each 'year' of play: the amount of land to be bought, the amount to be sold, the number of bushels of grain to be distributed to the people for food, and the number of acres of land

to be planted. Since the PLAYER program was essentially to take the place of a human player, the function of PLAYER was to provide these same four numeric values. PLAYER could, therefore, be considered a subroutine of the KINGDOM game, which would be called upon by KINGDOM at the beginning of each 'year' to provide the input values just as a human player would.

A human player generally makes 'yearly' decisions on the basis of the information presented to him by the KINGDOM program — e.g., how many bushels are currently in storage, how many people are in the kingdom, how many acres of land are owned, etc. Therefore, we assumed that the PLAYER program would have access to the same information, and thereby would be able to base its decisions

The following are absolute game constants:

FOOD — Each subject of the kingdom requires a minimum of 10 bushels of grain per year for use as food.

SEED — Each acre of land planted requires 3 bushels of grain per year for use as seed.

LABOR — Each subject may plant a maximum of 2 acres per year.

The following are variable factors:

LAND PRICE — (expressed as bushels per acre) — normally varies between 3 and 6 bushels per acre, unless affected by the occurrence of rain or drought. The harvest per acre for a given year becomes the trading value for the following year.

POPULATION — normally varies by 20 to 40% due to natural causes, unless affected by other natural occurrences.

GRAIN LOSS — from 0 to 50% of the harvest of a given year may be lost to rats. Only the harvest (not existing grain in storage) is susceptible to attack.

OVERFEEDING — a food surplus occurs when more grain is allotted for food than the minimal requirement. There is about a 50% probability of gaining one person for each 20 bushels over the minimum.

UNDERFEEDING — The ruler may starve off 2 persons per year without ill effect. Each starvation over 2 increases the probability of food riots by 5% each. A food riot may result in a loss of up to 50% of the grain in storage.

The following are purely random factors: (the probability of the event's occurrence in any given year is given in parenthesis):

PLAGUE — (10%) — may reduce population up to 80%.

THEFT — (10%) — may reduce grain storage by up to 10%.

EARTHQUAKE — (10%) — may reduce acreage by up to 15%.

ATTACK BY HUNS — (10%) — may reduce population up to 40%, storage up to 10% and land up to 2%.

RAIN — (15%) — increases harvest to 7, 8 or 9 bushels/acre.

DROUGHT — (15%) — decreases harvest to 1 or 2 bushels/acre.

GRAIN SHIPMENT — (15%) — increases grain storage — depends on the size of the kingdom (acres of land).

BORDER EXPANSION — (15%) — increases population and land — depends on present population of kingdom.

ions on these parameters. Also, like the human player, the PLAYER program would know the rules of the game and would restrict its play so as not to violate them. Having now defined the inputs and outputs for the PLAYER program, the only thing that remained was to define the process by which the program itself would simulate the 'decision process' of a human player. This, of course, required us to know just how the typical human KINGDOM player made these decisions, and was, perhaps, the most difficult phase of the entire RAMSES project.

To simulate human decision processes, we used a commonly accepted computer modeling technique in which the total decision process was broken down into a series of individual yes-no decisions,

throughout the course of the game, thereby enabling PLAYER to make most major decisions on the basis of the strategy it had been given.

Generally, decisions were not so well defined, however. Using the same example, a player's strategy might have read: 'Always sell land if the price is above 5 bushels per acre or if there is not enough grain in storage to feed all of the present population; but do not sell land at any price if there are too many people and some of them will deliberately have to starve this year.' Such a strategy obviously involves the examination of a number of parameters (storage, price, population, etc.) before the decision can be made, and the way in which each of these factors affects the decision has to be defined by yet another decision parameter.

Of course, making these yes-no decisions did not yield the numeric values which were to be generated by the PLAYER program. Once a decision is made to sell land, for example, the exact number of acres of land to be sold must be computed. Once again, this is a function of the current resources and the strategy being used by the player. A human player may say, 'If I am to sell land, sell all of my land if the price is above 8 bushels per acre, sell one-half of my land if the price is above 5 bushels per acre, but always save at least 25 acres in reserve or one acre for each person in the kingdom — whichever is greater.' Once again, this strategy involves a number of game values (land value, population, etc.) and may even include the making of another decision. In each case, the response value of the PLAYER program for the number of acres sold (corresponding to the response typed in to the KINGDOM program by a human player) was generated by a mathematical formula which included all of these factors in a manner consistent with the strategy being used. The defining factors in this formula which generated the response value were called *response parameters*, which, like the decision parameters, were entered into PLAYER prior to the start of the game and governed the actions of PLAYER throughout the game.

As we added more and more decision and response parameters to PLAYER, it became increasingly difficult to keep track of these parameters and to follow

the logical flow of the program. The task would have been totally impossible if we had not first laid out the program in the form of a gigantic flowchart, in which each decision, operation, and value was marked, and the flow of logic from one segment of the program to the next designated by a series of lines and arrows which, as time progressed, came to resemble either a large plate of spaghetti or a map of the New York subway system. This flowchart included not only the logic flow along with decisions and computations within the program, but also the variable names and copious notes.

As PLAYER was gradually developed and tested, other problems became apparent in terms of the decision modeling-process. First of all, when a human player plays KINGDOM, resource values will change as decisions are made; for example, if land is to be sold this 'year,' there will be less land to plant when the 'How much to plant?' question arises later. The human player would see this information as transactions occur; PLAYER, however, had to compute four response values at one time. Therefore, it was necessary to provide PLAYER with the capability of computing the results of its decisions; in effect, PLAYER had to foresee the results of its actions just as a human player could.

Also, some decisions made within the PLAYER program occasionally became factors in other later decisions. For example, if land was sold at a low price in order to provide food for the populace (say, to prevent starvation), PLAYER had to consider this when it came time to decide on buying land. In such a case, PLAYER could not buy land at the low price even though the strategy may normally have called for it to do so. Therefore, PLAYER had to be provided with a number of special internal indicators or 'flags' which were set under such special circumstances, and were tested later in the program to see if such an event had taken place. The portion of the program doing the testing had no way of knowing how the program arrived at this point. As there were so many complex paths to the program flow within PLAYER, such flags were essential in order for one part of PLAYER to keep track of what other parts were doing (the equivalent, perhaps, of a human player making notes to himself in order to remember the reasons behind particular decisions).

All of this process of programming, testing, adding, deleting, and changing elements of the PLAYER program filled our time schedules for most of a summer. During this time, we generated a stack of computer printout which could easily rival the Tower of Babel in altitude. We also found, perhaps, more ways to lose a game of KINGDOM than one could imagine was possible, and we became increasingly more respectful of that mysterious structure of the human mind which enabled a typical ten-year-old KINGDOM player to compute the decisions which our computer made only with the greatest of difficulty. Towards the end of the summer, PLAYER was finally perfected and went into full operation for the first time. There were



no less than 50 decision and response parameters which allowed us to define, we believed, any sort of KINGDOM strategy possible.

Still, not trusting the security of this incredibly complex program, we inserted a series of checks into the KINGDOM portion of the program which, in the case of an illegal move on the part of PLAYER, would immediately halt the entire program and print out a complete description of what had happened, along with all of the internal parameters, values, flag settings, and so forth, in order to trace down the fault and correct it. As time went on, we advanced from one error of this sort per day to one every other day, then one per week, then none at all.

The development of PLAYER proved to be invaluable in testing KINGDOM strategies given us by various human players and showed that a number of seemingly good strategies were actually poor performers in the long run. Likewise, what appeared to be a poor strategy would

occasionally prove very effective over the long run.

With PLAYER in working order, it was possible for RAMSES to play one game of KINGDOM every 5-10 seconds, giving our 7040 a 'testing rate' of about 500 games per hour on the average. Of course, strategies which lost games in short times ran much faster, and some strategies designed to produce long reigns would run considerably longer. Once we had computed all of the parameters which defined a given strategy, we would feed them into the computer, and effectively tell the computer to 'go play a few hundred games and tell us how this strategy works out.'

PLAYING KINGDOM AUTOMATICALLY

It was at this point in time that another phase of the RAMSES project came into existence: as long as RAMSES was playing all of these games for us, we might as well have the computer analyze the results also, thereby saving ourselves the trouble of doing this manually. It would also significantly reduce the volume of printout produced by RAMSES.

Thus, the MASTER program was born. The MASTER segment of RAMSES was made responsible for keeping track of game parameters, controlling the flow of information between KINGDOM and PLAYER, and most importantly, reducing the great volume of raw data produced by each series of games to a set of meaningful results. Test runs became standardized; each set of 100 games of KINGDOM was designated as a Dynasty, and MASTER would record and compute the statistics of each Dynasty such as the shortest win within the 100 games played, the longest winning reign, and various other statistics.

This was a definite improvement; what had been a six-inch stack of printout was now reduced to ten or twelve pages, with most (and eventually all) of the analysis being done by the computer rather than by us poor, overworked human beings. Later, we extended this approach, defining a new time span of an Era to be equivalent to 10 Dynasties, or 1000 games of KINGDOM. By feeding in a set of strategic parameters for 10 Dynasties, each slightly different from the last, the computer could then observe

what sort of change in performance took place as the strategy changed, and likewise analyze this data and reduce it to even more meaningful results. With this capability, it was possible to move one step closer to our goal of finding the 'ultimate strategy' for playing KINGDOM, as we could analyze not only a single strategy, but also examine a set of strategies which were closely related and choose the one which was best. In other words, we had a way by which to compute the *direction* in which to change our strategy to improve overall play of the game. Equally important, an average day's run of Eras now resulted in a much reduced pile of printout.

We spent about a month trying various strategies, modifying those that looked promising in the hopes they would lead us to the 'ultimate answer,' but alas, no 'ultimate answer' was forthcoming. It soon became obvious that there were so many variations of each strategy, that it would take us almost forever to trace down and evaluate each one in order to discover the true 'best' strategy — not to mention that we were searching not for one, but a complete set of 'best' strategies to cover a number of goals (fastest possible win, greatest number of wins, etc.). Once again, it occurred to us that perhaps the power of the computer could be used to do this for us, and this led eventually to the third, and final, phase of the RAMSES project.

LET YOUR COMPUTER DO THE SEARCHING

Before we begin a discussion of this third phase, it should be noted that most of this work is still in the planning stages; comparatively little has actually been implemented and tested using the computer. However, the little that actually was accomplished, plus the ideas devised for future work, warrant mentioning as part of the RAMSES project description.

The function of the third phase of the RAMSES project was, in essence, to develop a program which would not only play KINGDOM by a given strategy and evaluate that strategy, but would also be capable of modifying that strategy in an attempt to *improve* its play, and finally arrive at what would be the ultimate 'best strategy' for playing KINGDOM. In effect, the computer would 'learn' the game by trial and error, by testing

various strategies and choosing the one most effective in obtaining a defined goal. At first glance, this seemed to be relatively simple process to program into the machine. In PLAYER we already had a program to play the game of KINGDOM according to a given strategy, and in MASTER there existed a program to analyze the performance of the strategy being used. All that remained, therefore, was to write a program which changed strategy as the play proceeded and, as different strategies were tried, chose the best of the lot as the 'ultimate strategy.'

Unfortunately, it was not nearly as simple as it first appeared to be. To define a given strategy, as you recall, required the setting of over 50 decision and response parameters, each of which could assume



on the order of 10 *different* values (and some parameters would vary even more). This meant that, within the realm of the PLAYER program, there were on the order of *ten-to-the-fiftieth* possible strategies, each of which would require the playing of at least a complete Dynasty (100 games) to evaluate. We computed the amount of time it would have taken our computer to play this many games: ten-to-the-fifty-second seconds, or on the order of *1.5 times ten-to-the-forty-fifth years* of computer time! Obviously, another approach was called for.

Although we were not aware of it at the time we were doing this work, there is an entire field of mathematics which applies to such problems; this field is referred to most commonly as 'optimization theory.' This branch of mathematics deals with techniques of finding solutions to problems with a large number of possible solutions, and selecting from that number (by one means or another) the 'best' (or 'optimum') solution. Our first approach was, in the terminology of optimization theorists, an

'exhaustive search' approach, examining each possible solution and selecting the best. At first glance, this seemed to be a relatively simple process to program into the machine, however, as was proven above, it is not very practical (even through the use of high-speed computers) for big problems with *many* variables. There *are* other mathematical techniques for handling these types of problems; however, since we were not aware of them at the time, we proceeded to 'invent' our own.

What we did, in effect, was to return to the tactic of simulating the human player. As a human being plays the game, we noted that he not only modifies his strategy, but that he *learns from experience*. That is, if a certain change in strategy improves his play, he will continue to modify his strategy in that direction in the hopes of obtaining further improvement. On the other hand, if a modification in strategy makes for poorer overall play, the player abandons any further modification in that direction and proceeds to try new strategies in another direction.

From this observation, we formulated what we termed the 'up the hill' method of changing strategies. This seemingly odd name was derived from the example we used to demonstrate the method, which was as follows: suppose we imagine an area of land, a 'terrain' so to speak, which is not flat but is rather covered by hills and valleys. Each point in this terrain represents one particular KINGDOM strategy, and the elevation of the terrain shows the effectiveness of that strategy. The co-ordinates of that point, mathematically speaking, would be defined by the set of 50 or so parameters which define the strategy. From each point it is possible to move in any of 50 different directions (signifying a change in strategy) which could have one of three possible results: we could go higher (improving the play), lower (degrading the play) or we could remain at the same altitude (no change). Our goal is to make our way to the highest peak in the landscape, at which point the 'best' strategy exists.

We are blindfolded, so we cannot simply look around, climb the nearest hill and arrive at the top. Rather, we do the next best thing — we 'feel around' to determine which directions lead upward, pick the steepest path, and follow that path as

long as it leads upward. When we have climbed as high as is possible along that path, we search for another direction which leads upward and follow that path, etc. If we can find no direction which leads further upward, we must be at the peak of a hill, which indicates at least some 'highest point' has been found (but not necessarily *the highest* point in the land). We note the coordinates, choose another starting point, and go searching again. Providing our starting points are far enough apart, we will eventually find all of the 'local high points,' and from them, select the highest as the 'ultimate answer.'

This is essentially the means by which we had hoped to design the RAMSES program to solve the game of KINGDOM. We would program it to start at some defined strategy, and it would play through a Dynasty or two using PLAYER, then evaluate the performance of this strategy through MASTER. It would then, in turn, change each of the strategy parameters a small amount, play more Dynasties, and evaluate each new strategy for effectiveness. Then, the computer would select the parameter whose change was responsible for the *greatest* improvement in performance, and continue to change that parameter in the chosen direction until no further improvement was detected in play. At this time, the computer would once again 'feel around' for another upward path, and follow it to improve play even more. This process would occur over and over, 'tuning' each parameter in turn, always improving play, until the 'summit' was reached. At this point, the parameter would be recorded, along with the overall evaluation of play with this strategy, and the entire process would start again from a *new* starting point.

All of this involves a great deal of rather complex programming, and for the most part, is still in the development phase. There is also the problem of selecting starting points; for this, we would use either randomly selected values, or perhaps we would start the program with a strategy first developed by human players and have the computer try to 'improve on it.' Such a process would still involve the use of much computer time, and the playing of many Dynasties and Eras, but would require enormously less time than trying to evaluate *every* possible strategy available.

The few results obtained from the work completed were at least interesting, if not totally conclusive. Tests showed some parameters affected the play of the game greatly; these were generally parameters which controlled the population of the kingdom (which was to be expected, since population is the critical parameter in KINGDOM). Other parameters had little or no effect on most strategies; for example, the percentage of land planted in a given year would often prove to have no long-range effect on the course of the game. The computer, like many human players, indicated that the population initially assigned at the beginning of the game was too large, and further indicated that deliberate reduction of population was necessary, at least somewhat to



assure a winning game. (Since this was a deliberate 'fix' in the KINGDOM program, it would have been rather surprising if strategies ignoring the 'fix' were successful; however, one never knows.)

During this time of testing and development, we became aware of a number of new items previously never considered. There were some factors in human play which had never been included in PLAYER, and which perhaps, could never be included in a computer program. A human player uses 'intuition' in playing the game, and would often 'play the odds' in hoping that rain would come or a grain shipment would arrive in the kingdom. Perhaps the computer could be programmed to keep count of such random occurrences and 'play the odds' by attempting to predict them as the human player does, but is this really the same thing?

Also, a human player does not always play by a fixed strategy as the game progresses, but may vary his mode of play as time goes on. It is known, for example, that in the beginning of a game of KINGDOM, planting and harvesting is a major

source of income, but as the game progresses and wealth builds up, the harvest becomes a relatively unimportant factor. Would it be possible to develop a computer program to play by such a dynamic strategy? We think it may be possible, by including time as an element in the strategy parameters, but as of yet we have not developed a program to do so.

It was not until two years after the start of the RAMSES project that we discovered our 'up-the-hill' technique already existed. This technique is commonly known to optimization theorists as a 'gradient search' method, involving the use of complex mathematics (rather than 'feeling around') to determine the direction of steepest ascent and the distance to travel. There was, we discovered, even a complete field of optimization theory designated as 'stochastic programming' which is used to analyze problems dealing in uncertainties and random events such as those which occur in the game of KINGDOM. Such forms of analysis may even make unnecessary the playing of large numbers of games to 'even out' random factors, but instead include them as part of the mathematical calculations.

We do not feel, however, that our efforts on the RAMSES project were wasted, even though there is probably a mathematical method somewhere which could possibly find the optimal 'best' strategy for us. On the contrary, we feel we learned much from this project, and we are learning even more now as the project continues to this day. Perhaps someday we will investigate the totally mathematical approach; however, for now, we will continue to tinker with RAMSES and KINGDOM as time permits, and perhaps someday RAMSES will discover the strategy which gives the player the fastest, the slowest, or the most certain win. Until that time, however, we have piles and piles of printout to scratch notes on the back of, the world's record in number of games of KINGDOM played by a single group, and we still can sit down at the computer terminal and rule the world whenever that mysterious urge passes over us. It is perhaps comforting to know that, despite all our efforts to solve this game programmatically, KINGDOM remains as one of the few endeavours in which a human being can outperform a computer, even if it is in the pursuit of such a trivial goal as ruling the entire world.

Listing

```
10 REM "KINGDOM" FOR TSS/8 BASIC
11 REM
12 REM VERSION 1 MOD 2 12/28/73
13 REM
14 REM
15 REM WRITTEN BY:
16 REM T. VOKOS A.O. SMITH CORP.
17 REM L. SCHNEIDER UWM, MILWAUKEE
18 REM
19 REM
20 RANDOMIZE
21 DEF FNR(Z1)=INT(IN1(Z1)*RND(0))
22 DEF FNL(Z2)=FNR(100)-Z2
23 REM
24 REM ***SET INITIAL PARAMETERS***
25 REM
26 LET Y=0
27 LET L0=10*6
28 LET L1=3
29 LET N0=FNR(75)+75
30 LET L2=FNR(250)+250
31 LET N1=FNR(3000)+2000
32 LET L3=0
33 LET N2=0
34 PRINT
35 REM
36 REM ***PRINT YEARLY REPORT***
37 REM
38 PRINT "<<<< >>>> <<<< >>>> <<<< >>>> <<<< >>>> <<<< >>>>"
39 PRINT
40 PRINT "REPORT FOR YEAR "Y
41 PRINT
42 PRINT "POPULATION IS "N0
43 PRINT "ACRES OF LAND OWNED "L2
44 PRINT "BUSHELS IN STORAGE "N1
45 PRINT "PRICE OF LAND IS "L1" BUSHELS PER ACRE"
46 PRINT
47 REM
48 REM ***READ AND VERIFY LAND TRANSACTIONS***
49 REM
50 PRINT "HOW MANY ACRES TO BUY?"
51 INPUT B
52 LET B=INT(B)
53 IF B<0 GOTO 2020
54 IF B=0 GOTO 270
55 LET A=N1-B*L1
56 IF A<0 GOTO 260
57 PRINT "YOUR STORAGE IS ONLY "N1" BUSHELS!"
58 GOTO 215
59 LET N1=A
60 LET L2=L2+B
61 PRINT "TO SELL?"
62 INPUT C
63 LET C=INT(C)
64 IF C<0 GOTO 2020
65 IF C=0 GOTO 325
66 LET A=L2-C
67 IF A<0 GOTO 315
68 PRINT "YOU ONLY OWN "L2" ACRES!"
69 GOTO 270
70 LET L2=A
71 LET N1=N1+C*L1
72 IF (B+C)=0 GOTO 335
73 PRINT "TRANSACTION RESULTS: LAND "L2" ACRES;
GRAIN "N1" BUSHELS"
74 REM
75 REM ***TEST FOR WIN***
76 REM
77 IF (L2+N1-L0)>=0 GOTO 2010
78 IF N1=0 GOTO 490
79 REM
80 REM ***READ AND VERIFY FOOD AND SEED ALLOTMENT***
81 REM
82 PRINT
83 PRINT "HOW MANY BUSHELS FOR FOOD?"
84 INPUT N2
85 N2=INT(N2)
86 IF N2<0 GOTO 2020
87 LET A=N1-N2
88 IF A<0 GOTO 390
89 PRINT "YOUR STORAGE IS ONLY "N1" BUSHELS!"
90 GOTO 350
91 LET N1=A
92 IF N1=0 GOTO 490
93 PRINT "HOW MANY ACRES TO BE PLANTED?"
94 INPUT L3
95 LET L3=INT(L3)
96 IF L3<0 GOTO 2020
97 IF L3=0 GOTO 500
98 IF (L2-L3)>=0 GOTO 440
99 PRINT "YOU ONLY OWN "L2" ACRES!"
100 GOTO 400
101 IF (2*N0-L3)>=0 GOTO 455
102 PRINT "YOUR POPULATION IS ONLY "N0" PEOPLE!"
103 GOTO 400
104 LET A=N1-3*L3
105 IF A<0 GOTO 475
106 PRINT "YOUR STORAGE IS ONLY "N1" BUSHELS!"
107 GOTO 400
108 LET N1=A
```

```
488 IF N1>0 GOTO 500
489 PRINT
490 PRINT "STORAGE IS NOW EMPTY...GOOD LUCK!"
491 REM
492 REM ***COMPUTE EFFECTS OF FOOD ALLOTMENT***
493 REM
494 LET Y=Y+1
495 LET N3=FNR(N0/3+3)
496 LET N4=FNR(N0/4+2)
497 LET N5=N0-IN1(N2/10)
498 IF N5>0 GOTO 560
499 IF N5=0 GOTO 590
500 LET A=FNR(3-N5/2)
501 PRINT
502 PRINT "***FOOD SURPLUS*** POPULATION INCREASE "Y
503 LET N5=0
504 LET N3=N3+A
505 GOTO 590
506 IF (FNL(5+(N5-2)))>0 GOTO 590
507 LET A=FNR((N5*N1)/(2*N0))
508 PRINT
509 PRINT "***FOOD KLOTS*** "Y" BUSHELS LOST"
510 LET N1=N1-A
511 IF FNL(10)>0 GOTO 620
512 LET A=INT(N0/3)+FNR(N0/2+2)
513 REM
514 REM ***COMPUTE RANDOM OCCURANCES***
515 REM
516 PRINT
517 PRINT "***LAGUE*** "Y" DIED"
518 LET N4=N4+A
519 IF FNL(10)>0 GOTO 670
520 LET A=FNR(N0/5)+INT(N0/5)
521 LET B=FNR(N1/20)+INT(N1/20)
522 LET C=FNR(L2/50)
523 PRINT
524 PRINT "***HUNS ATTACK***"
525 PRINT A" PEOPLE KILLED "B" BUSHELS TAKEN "C" ACRES DESTROYED"
526 LET N4=N4+A
527 LET N1=N1-B
528 LET L2=L2-C
529 IF FNL(15)>0 GOTO 705
530 LET A=FNR(20)+10
531 LET B=FNR(N0+250)+50
532 PRINT
533 PRINT "***BOXDER EXPANSION*** YOU GAINED "Y" PEOPLE; "B" ACRES"
534 LET N3=N3+A
535 LET L2=L2+B
536 LET N0=N0+N3-N4-N5
537 IF N0<=0 GOTO 2015
538 PRINT
539 PRINT N3" ARRIVED"
540 PRINT N5" DIED OF STARVATION"
541 PRINT N4" DIED NATURAL CAUSES"
542 IF FNL(10)>0 GOTO 740
543 LET A=FNR(N1/20)+INT(N1/20)
544 PRINT
545 PRINT "***THEFT*** "Y" BUSHELS STOLEN"
546 LET N1=N1-A
547 IF FNL(10)>0 GOTO 785
548 LET A=FNR(L2/10)+INT(L2/20)
549 PRINT
550 PRINT "*****EARTHQUAKE***** "Y" ACRES DESTROYED"
551 LET L2=L2-A
552 IF FNL(15)>0 GOTO 810
553 LET A=FNR(100)+INT(L2/100)+500
554 PRINT
555 PRINT "***GRAIN SHIPMENT ARRIVES*** "Y" BUSHELS"
556 LET N1=N1+A
557 IF FNL(15)>0 GOTO 835
558 LET L1=FNR(2)+1
559 PRINT
560 PRINT "***DROUGHT***"
561 GOTO 860
562 IF FNL(15)>0 GOTO 852
563 LET L1=FNR(3)+7
564 PRINT
565 PRINT "***RAIN***"
566 GOTO 860
567 LET L1=FNR(4)+3
568 REM
569 REM ***COMPUTE HARVEST***
570 REM
571 LET A=L1*L3
572 LET B=FNR(A/2)
573 LET C=A-B
574 PRINT
575 PRINT
576 PRINT
577 IF A=0 GOTO 910
578 PRINT "HARVEST WAS "L1" BUSHELS PER ACRE FOR A TOTAL "Y" BUSHELS"
579 PRINT "LOSS TO KATS "B" BUSHELS; NET HARVEST WAS "C" BUSHELS"
580 LET N1=N1+C
581 GOTO 130
582 PRINT "HARVEST WAS "L1" BUSHELS PER ACRE FOR A TOTAL "Y" BUSHELS"
583 GOTO 900
584 REM
585 REM ***SPECIAL MESSAGES FOR ENDINGS***
586 REM
587 PRINT
588 PRINT
589 PRINT "***CONGRATULATIONS*** YOU NOW RULE THE WORLD!"
590 GOTO 2020
591 PRINT
592 PRINT "***DISASTER*** - THERE ARE NO MORE PEOPLE LEFT!"
593 PRINT
594 PRINT "YOUR REIGN LASTED "Y" YEARS"
595 REM
596 END
```


SNOOPING WITH YOUR PET

A Sophisticated Guide to PEEKing and POKEing Around



BY MARK ZIMMERMANN

Mark Zimmermann is a graduate student at Caltech, where he does work mainly on stars and gravitation. Since February, he's added an 8K PET to his list of interests. —Phyllis Cole

If you're like I am, and want to make full use of the Commodore PET's machine language abilities, a few discoveries and programming aids that I've put together may help you. Users of other 6502-based systems should also find a lot of this information valuable, as should anyone blessed with a BASIC interpreter from Microsoft. I have not seen any of this information published elsewhere; I found it by peeking, poking and mapping various pages of memory to the screen.

First, the PET's number system: base 2, excess 128, floating point numbers with 32 (binary) digits. Donald Knuth in chapter 4 of his monumental series *The Art of Computer Programming* gives an excellent discussion of floating point arithmetic,

and the above description is in his terminology. What it means is simple.

Consider a floating point binary number, for example, $3_{10} = 11_2 = 2^2 \times 0.11_2$. Note that we have 'normalized' the floating point representation by factoring out powers of 2 (that is, moving the radix point*) to get a number with all zeroes to the left of the point, and a one immediately to the right.

The 'exponent' for this number is the power of two that appears—namely 2. The 'fraction' part is 0.11000...

'Base 2' means that we work in binary. 'Excess 128' means that to avoid storing negative exponents, we add 128 to the power of two that usually appears. '32 digits' means that we store 4 bytes, each of 8 bits, containing the fractional part of the number.

*It's not a decimal point!!

Specifically, suppose that in memory we store the exponent and the four fractional bytes in order, as E, F1, F2, F3, F4. Then the number 3_{10} above would be represented by $E = 2 + 128 = 130_{10}$, $F1 = 11000000_2 = 192_{10}$, $F2 = F3 = F4 = 0$.

The PET does this, with one clever modification. You may have noticed that by our definition of normalization, the fractional part of every nonzero number has a one immediately to the right of the radix point. So, the '128's digit,' bit 7, of F1 is always 1 unless the number represented is 0. That's a pretty inefficient use of the bit.

You may also have noticed that we forgot to indicate the sign of the number. What if we want to represent a negative number like -3?

The designers of the PET's BASIC solved both problems by using bit 7 of F1 as the sign bit. That bit is 0 if the number is positive, and 1 if negative, and the 1 which should always begin F1 is implied. (Knuth suggests exactly this trick in exercise 2, section 4.2.1 of his book.)

So, what will you see if you PEEK into a region of the PET's memory containing 3_{10} ? You'll find the sequence 130, 64, 0, 0, 0. If π is stored there, you'll see 130, 73, 15, 281, 161, to give a less trivial example.

It's not hard to convert to and from this PET format. Program A takes E, F1, F2, F3, F4, and makes N, the number that they represent. Program B accepts N as input and produces E, F1, etc. The programs are written in PET BASIC, but should be easy to adapt to any other BASIC which keeps enough significant figures during calculations. On a PET itself, it is possible that accumulated roundoff errors may change the least significant digit of N or F4, so beware. Note the convention used to represent zero: a stored set of 0, 0, 0, 0.

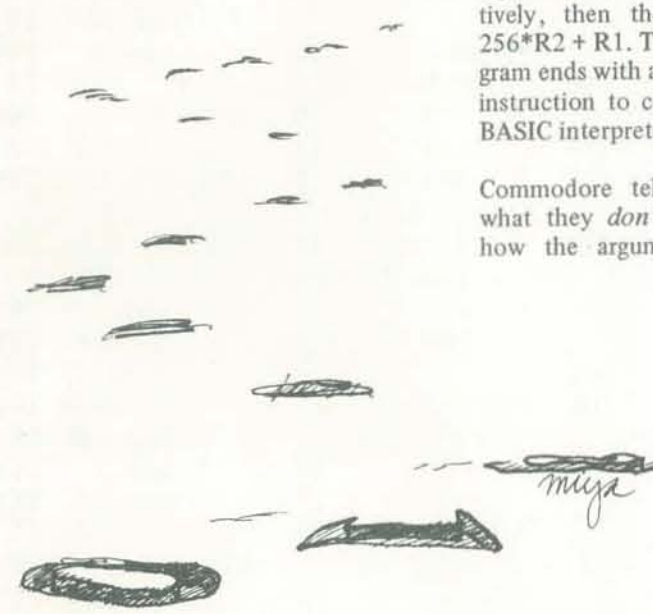
Another fascinating feature of the PET's BASIC is the USR(X) function, which when called (as in '10 A=USR(B)') jumps to a machine-language subroutine at an address stored by the user in memory locations 1 and 2. If the contents of these registers are called R1 and R2 respectively, then the address jumped to is $256 \times R2 + R1$. The machine-language program ends with a 'return from subroutine' instruction to come back to running the BASIC interpreter.

Commodore tells you that much, but what they *don't* tell you is where and how the argument X of the function

USR(X) is stored. To find that took me quite a lot of peeking around! It turns out that page zero, memory locations 176 through 181 is the place. Memory 176 contains E, 178 contains F2, 179 has F3, and 180 has F4. In 177 is F1, BUT the sign bit has been changed to a 1, making this a true binary-floating-point-excess-128 number. The sign bit is bit 7 or memory location 181.

To return a value different from the X that went into USR(X), the user's machine language program must put that number into registers 176 to 181 in the format described.

Compared to floating point, 'integer' variables like X% and YZ% are stored in a very simple format. The interpreter allocates 5 bytes (plus 2 for the variable name) as for the floating point number, but only uses the first 2 of them. Call them J1 and J2. J1 represents a positive number from 0 to 127 if its bit 7 is zero; if bit 7 is one, J1 is a negative number in two's complement notation. (For example, $-1 = 11111111$, $-2 = 11111110$, ..., $-128 = 10000000$. This is the convention used by the 6502 chip itself in calculating branching addresses.) J2 is an ordinary number from 0 to 255 in binary. The in-



Program A

```

10 REM PET TO USUAL CONVERSION
20 REM COPYRIGHT 1978 MARK ZIMMERMANN
30 REM PERMISSION TO USE, NOT TO SELL
40 INPUT E,F1,F2,F3,F4
50 S=1: REM SIGN OF N
60 IF F1>=128 THEN S=-1: F1=F1-128
65 F5=(((F4/256+F3)/256+F2)/256+F1)/256+.5)
70 N=2↑(E-128)*F5*S
80 PRINT N
90 END
    
```

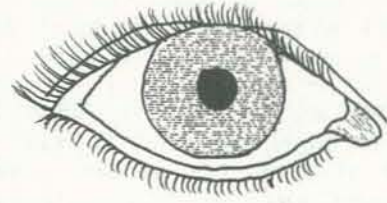
Program B

```

10 REM USUAL TO PET NUMBER CONVERSION
20 REM COPYRIGHT 1978 MARK ZIMMERMANN
30 REM PERMISSION TO USE, NOT TO SELL
40 INPUT N
50 IF N=0 THEN E=0:F1=0:F2=0:F3=0:F4=0:END
60 REM PET CONVENTION FOR ZERO
70 S=0:REM SIGN BIT OF F1
80 IF N<0 THEN S=1:N=-N
90 E=128
100 IF N>=1 THEN E=E+1:N=N/2:GOTO 100
110 IF N<.5 THEN E=E-1:N=N*2:GOTO 110
120 REM NOW E IS RIGHT & N IS NORMALIZED
130 N=N-.5:REM CLEAR SIGN BIT
140 F1=INT(N*256)
150 N=256*N-F1
160 F1=F1+128*S:REM INSERT SIGN BIT
170 F2=INT(256*N)
180 N=256*N-F2
190 F3=INT(256*N)
200 N=256*N-F3
210 F4=INT(256*N)
220 PRINT E:F1:F2:F3:F4
230 END
    
```

teger represented by J1, J2 has value 256*J1+J2. In a call to USR(X%), however, the number is converted to the usual true floating point binary, excess 128 format for bits 176 - 180, with sign still in bit 7 of 181. So, integer variables aren't all that relevant to using USR(X).

Finally, for a more mundane but tremendously useful programming aid, see Tables 1 and 2. They contain *decimal* op codes and mnemonics, an absolute necessity for PET peekers and pokers. For



some unknown reason, these tables never seem to be published in such a usable format. The information in them is derived from the excellent 'MCS6500 Microcomputer Family Programming Manual,' published by MOS Technology, Inc. The mnemonics used in the tables are all clearly defined and explained there (in hexadecimal, damn it!). I've only been programming in machine language for a couple of weeks, and the decimal op code tables I laboriously compiled for myself have been invaluable. Enjoy!

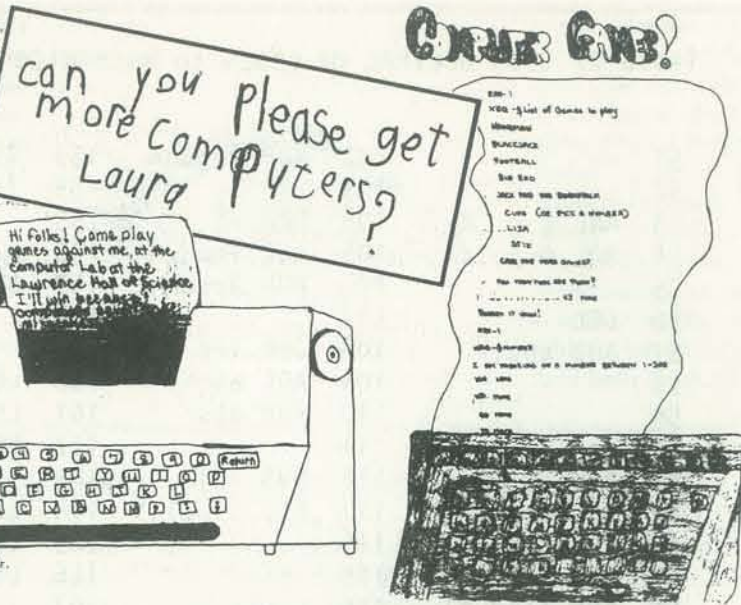
Table 1: 650X MNEMONICS to DECIMAL OP CODES conversion

| | | | | | | | | | |
|---------|-----|---------|-----|---------|-----|---------|-----|---------|-----|
| ADC imm | 105 | BVC rel | 80 | 0 page | 69 | LSR abs | 78 | abs | 237 |
| abs | 109 | | | (ind,X) | 65 | 0 page | 70 | 0 page | 229 |
| 0 page | 101 | BVS rel | 112 | (ind),Y | 81 | accum | 74 | (ind,X) | 225 |
| (ind,X) | 97 | | | 0 p.,X | 85 | 0 p.,X | 86 | (ind),Y | 241 |
| (ind),Y | 113 | CLC | 24 | abs,X | 93 | abs,X | 94 | 0 p.,X | 245 |
| 0 p.,X | 117 | | | abs,Y | 89 | | | abs,X | 253 |
| abs,X | 125 | CLD | 216 | | | NOP | 234 | abs,Y | 249 |
| abs,Y | 121 | | | INC abs | 238 | | | | |
| | | CLI | 88 | 0 page | 230 | ORA imm | 9 | SEC | 56 |
| | | | | 0 p.,X | 246 | abs | 13 | | |
| AND imm | 41 | CLV | 184 | abs,X | 254 | 0 page | 5 | SED | 248 |
| abs | 45 | | | | | (ind,X) | 1 | | |
| 0 page | 37 | | | | | (ind),Y | 17 | SEI | 120 |
| (ind,X) | 33 | CMP imm | 201 | INX | 232 | | | | |
| (ind),Y | 49 | abs | 205 | | | 0 p.,X | 21 | | |
| 0 p.,X | 53 | 0 page | 197 | INY | 200 | abs,X | 29 | STA abs | 141 |
| abs,X | 61 | (ind,X) | 193 | | | abs,Y | 25 | 0 page | 133 |
| abs,Y | 57 | (ind),Y | 209 | JMP abs | 76 | | | (ind,X) | 129 |
| | | 0 p.,X | 213 | ind | 108 | PHA | 72 | (ind),Y | 145 |
| ASL abs | 14 | abs,X | 221 | | | | | 0 p.,X | 149 |
| 0 page | 6 | abs,Y | 217 | JSR abs | 32 | PHP | 8 | abs,X | 157 |
| accum | 10 | | | | | | | abs,Y | 153 |
| 0 p.,X | 22 | CPX imm | 224 | LDA imm | 169 | PLA | 104 | | |
| abs,X | 30 | abs | 236 | abs | 173 | | | STX abs | 142 |
| | | 0 page | 228 | 0 page | 165 | PLP | 40 | 0 page | 134 |
| | | | | (ind,X) | 161 | | | 0 p.,Y | 150 |
| BCC rel | 144 | | | (ind),Y | 177 | | | | |
| | | CPY imm | 192 | 0 p.,X | 181 | ROL abs | 46 | | |
| | | abs | 204 | abs,X | 189 | 0 page | 38 | STY abs | 140 |
| | | 0 page | 196 | abs,Y | 185 | accum | 42 | 0 page | 132 |
| | | | | | | 0 p.,X | 54 | 0 p.,X | 148 |
| BEQ rel | 240 | | | abs,X | 62 | | | | |
| | | DEC abs | 206 | LDX imm | 162 | | | | |
| | | 0 page | 198 | abs | 174 | ROR abs | 110 | TAX | 170 |
| BIT abs | 44 | | | 0 page | 166 | 0 page | 102 | | |
| 0 page | 36 | | | 0 p.,X | 190 | accum | 106 | TAY | 168 |
| | | | | abs,X | 182 | 0 p.,X | 118 | | |
| BMI rel | 48 | DEX | 202 | 0 p.,Y | 182 | abs,X | 126 | TSX | 186 |
| | | | | | | | | | |
| BNE rel | 208 | DEY | 136 | LDY imm | 160 | | | TXA | 138 |
| | | | | abs | 172 | RTI | 64 | | |
| | | | | 0 page | 164 | | | | |
| BPL rel | 16 | | | 0 p.,X | 180 | RTS | 96 | TXS | 154 |
| | | EOR imm | 73 | abs,X | 188 | SBC imm | 233 | | |
| BRK | 0 | abs | 77 | | | | | TYA | 152 |

Table 2: 650X DECIMAL OP CODES to MNEMONICS conversion

| | | | | | | | | | |
|----|-------------|-----|-------------|------------|-------------|-----------|-------------|---------|-------------|
| 0 | BRK | 51 | 102 | ROR 0 page | 153 | STA abs,Y | 204 | CPY abs | |
| 1 | ORA (ind,X) | 52 | 103 | | 154 | TXS | 205 | CMP abs | |
| 2 | | 53 | AND 0 p.,X | 104 | PLA | 155 | 206 | DEC abs | |
| 3 | | 54 | ROL 0 p.,X | 105 | ADC imm | 156 | 207 | | |
| 4 | | 55 | | 106 | ROR accum | 157 | STA abs,X | 208 | BNE rel |
| 5 | ORA 0 page | 56 | SEC | 107 | | 158 | | 209 | CMP (ind),Y |
| 6 | ASL 0 page | 57 | AND abs,Y | 108 | JMP ind | 159 | | 210 | |
| 7 | | 58 | | 109 | ADC abs | 160 | LDY imm | 211 | |
| 8 | PHP | 59 | | 110 | ROR abs | 161 | LDA (ind,X) | 212 | |
| 9 | ORA imm | 60 | | 111 | | 162 | LDX imm | 213 | CMP 0 p.,X |
| 10 | ASL accum | 61 | AND abs,X | 112 | BVS rel | 163 | | 214 | DEC 0 p.,X |
| 11 | | 62 | ROL abs,X | 113 | ADC (ind),Y | 164 | LDY 0 page | 215 | |
| 12 | | 63 | | 114 | | 165 | LDA 0 page | 216 | CLD |
| 13 | ORA abs | 64 | RTI | 115 | | 166 | LDX 0 page | 217 | CMP abs,Y |
| 14 | ASL abs | 65 | EOR (ind,X) | 116 | | 167 | | 218 | |
| 15 | | 66 | | 117 | ADC 0 p.,X | 168 | TAY | 219 | |
| 16 | BPL rel | 67 | | 118 | ROR 0 p.,X | 169 | LDA imm | 220 | |
| 17 | ORA (ind),Y | 68 | | 119 | | 170 | TAX | 221 | CMP abs,X |
| 18 | | 69 | EOR 0 page | 120 | SEI | 171 | | 222 | DEC abs,X |
| 19 | | 70 | LSR 0 page | 121 | ADC abs,Y | 172 | LDY abs | 223 | |
| 20 | | 71 | | 122 | | 173 | LDA abs | 224 | CPX imm |
| 21 | ORA 0 p.,X | 72 | PHA | 123 | | 174 | LDX abs | 225 | SBC (ind,X) |
| 22 | ASL 0 p.,X | 73 | EOR imm | 124 | | 175 | | 226 | |
| 23 | | 74 | LSR accum | 125 | ADC abs,X | 176 | BCS rel | 227 | |
| 24 | CLC | 75 | | 126 | ROR abs,X | 177 | LDA (ind),Y | 228 | CPX 0 page |
| 25 | ORA abs,Y | 76 | JMP abs | 127 | | 178 | | 229 | SBC 0 page |
| 26 | | 77 | EOR abs | 128 | | 179 | | 230 | INC 0 page |
| 27 | | 78 | LSR abs | 129 | STA (ind,X) | 180 | LDY 0 p.,X | 231 | |
| 28 | | 79 | | 130 | | 181 | LDA 0 p.,X | 232 | INX |
| 29 | ORA abs,X | 80 | BVC rel | 131 | | 182 | LDX 0 p.,Y | 233 | SBC imm |
| 30 | ASL abs,X | 81 | EOR (ind),Y | 132 | STY 0 page | 183 | | 234 | NOP |
| 31 | | 82 | | 133 | STA 0 page | 184 | CLV | 235 | |
| 32 | JSR abs | 83 | | 134 | STX 0 page | 185 | LDA abs,Y | 236 | CPX abs |
| 33 | AND (ind,X) | 84 | | 135 | | 186 | TSX | 237 | SBC abs |
| 34 | | 85 | EOR 0 p.,X | 136 | DEY | 187 | | 238 | INC abs |
| 35 | | 86 | LSR 0 p.,X | 137 | | 188 | LDY abs,X | 239 | |
| 36 | BIT 0 page | 87 | | 138 | TXA | 189 | LDA abs,X | 240 | BEQ rel |
| 37 | AND 0 page | 88 | CLI | 139 | | 190 | LDX abs,Y | 241 | SBC (ind),Y |
| 38 | ROL 0 page | 89 | EOR abs,Y | 140 | STY abs | 191 | | 242 | |
| 39 | | 90 | | 141 | STA abs | 192 | CPY imm | 243 | |
| 40 | PLP | 91 | | 142 | STX abs | 193 | CMP (ind,X) | 244 | |
| 41 | AND imm | 92 | | 143 | | 194 | | 245 | SBC 0 p.,X |
| 42 | ROL accum | 93 | EOR abs,X | 144 | BCC rel | 195 | | 246 | INC 0 p.,X |
| 43 | | 94 | LSR abs,X | 145 | STA (ind),Y | 196 | CPY 0 page | 247 | |
| 44 | BIT abs | 95 | | 146 | | 197 | CMP 0 page | 248 | SED |
| 45 | AND abs | 96 | RTS | 147 | | 198 | DEC 0 page | 249 | SBC abs,Y |
| 46 | ROL abs | 97 | ADC (ind,X) | 148 | STY 0 p.,X | 199 | | 250 | |
| 47 | | 98 | | 149 | STA 0 p.,X | 200 | INY | 251 | |
| 48 | BMI rel | 99 | | 150 | STX 0 p.,Y | 201 | CMP imm | 252 | |
| 49 | AND (ind),Y | 100 | | 151 | | 202 | DEX | 253 | SBC abs,X |
| 50 | | 101 | ADC 0 page | 152 | TYA | 203 | | 254 | INC abs,X |
| | | | | | | | | 255 | □ |

100-HELLO
WHAT IS YOUR NAME?
HELLO AM, TELL ME ALL ABOUT YOUR
IF AN HAVING TROUBLE WRITING THIS A
YOU LONG HAVE YOU BEEN HAVING TROUBLE
SINCE I STARTED, ABOUT THE MOTHER
WHAT NAME CAN YOU TELL ME ABOUT THE
YOUR ARTICLE IS ABOUT PUBLIC ACCESS
WHY DO YOU MENTION COMPUTERS?
COMPUTERS ARE VERY USEFUL. TELL ME
WHAT DO YOU THINK MACHINES HAVE
COMPUTERS HAVE A MAN PUBLIC IN
DON'T YOU THINK COMPUTERS CAN DO
YES, OF COURSE.
PLEASE TELL ME MORE.
PERSONAL COMPUTING IS "THINK
PLEASE DO ON.
WE MUST FIND WAYS TO INTRODUCE THE
TELL ME MORE.
WE MUST GIVE THE PUBLIC HANDS
WHAT ABOUT MACHINES WORRIES
SHOWING INTRINSICALLY FROM
WHAT DOES THAT SUGGEST TO YOU?
WHAT WE NEED TO EDUCATE THE
WHAT DO YOU THINK ABOUT MAKING
THEY ARE TOOLS FOR EXERCISE
WOULD YOU LIKE IT IF THE
T
WHAT IS A SILLY GAME?
DO YOU SAY THAT IS A
DRAW FOR SOME SPECIAL
YES... I AM REMEMBERING
DO YOU ENJOY BEING UP
SI AM ECSTATIC! THIS IS
IS IT BECAUSE YOU ARE
CAME TO ME?
YES, I AM AFRAID SO.
HOW LONG HAVE YOU BEEN
NOW, FOR ABOUT TEN MIN
DO YOU FEEL STRONGLY
WHAT THINGS
DOES THAT OUTFITTING IN
DO YOU SAY WE HAVE
ONE SPECIAL REASON?
SI REALLY HAVE TO
THAT'S QUITE INTERESTING
YOU MUST STOP THE
OH, I MUST STOP THE
YES, STOP THIS CONVERSATION AT ONCE.
WHAT ELSE CAN YOU TELL ME?
*STOP, I SAY!
THANKS FOR TALKING TO ME. BYE BYE.



Computers and Science Museums: A Public Access Model

BY BOB KAHN

PART II

To achieve fully the goal of public computer literacy, we must go beyond books, magazines and films about computers; people must have hands-on access to computers in a personal context. How can such public access be made available? Where, besides computer fairs and stores, can people go to learn about and play with computers? One answer: a number of science museums or science-technology centers all across the country. This is the second of a three-part serialized article exploring the ways in which these centers are making computers accessible and understandable to the general public. An earlier version of this article was published in IEEE Computer Magazine, April 1977. We are grateful to IEEE for granting us permission to reprint sections of the original article and, in particular, the figures and tables. —BK

In the first part of this article, which appeared in the last issue of People's Computers, I discussed the Ground Level and Level I of the ladder-like model shown in Figure 1. This model describes the involvement of many science-technology centers in making computers accessible and understandable to the public.

In Part I, I noted that the Ground Level, in which a museum acquires computing power to meet its information and data processing need, does not necessarily represent public access to computers, but that an institution using computers for this purpose can potentially begin to explore ways to make computing available to the public.

Based on material originally appearing in 'Public Access to Personal Computing: A New Role for Science Museums' from *COMPUTER Magazine*, April, 1977, Reprinted with permission of IEEE.

At level I computers are available to special groups—usually children—but access is restricted: either a very few special children learn a great deal about computers or a great many children have a very brief introduction, often with no avenue for follow-up. The model places more value on programs that encourage follow-up opportunities so that once people have become more interested in computers, they will have an opportunity to learn more about them and they will have a place where they can go to use them. Also, the model is based on access for the general public which includes adults as well as children, and a broad spectrum of people rather than selected groups.

In Part II of this article, I now describe Level II of the model which shows a definite step upward in regard to both of these points. Indeed, Level II represents a realistic goal towards which a large number of science-technology centers are currently striving.

LEVEL II: GENERAL PUBLIC ACCESS

Public Fascination and Museum Commitment: A scenario. On any given weekend, one may walk into any of the institutions listed in Levels II and III of Figure 1 and observe roughly the same phenomenon: There will always be a group of involved, happy, energetic children and some adults huddled around the computer terminal(s)... usually playing games. (This is assuming that the terminal and/or the computer is up and running; if it is not, change the scenario to read "... there will always be a group of children pounding on the terminal(s), writing notes to the suggestion box or complain-

| Levels | Characteristics | Examples |
|------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| LEVEL IV: Ideal Public Access Computing Center | Basic Research on the Person-Computer Interaction Development of New Personal and Public Uses of Computers Community Program Library and Data Bank Publicly Accessible Showcase for Newest Technology Development and Dissemination of Educational Computer Programs, Resource for Personal & Home Computing Needs | |
| LEVEL III: Active Community Computer Education | Community Computer Education Resource and Time-Sharing Service Range of Classes for Children & Adults Many Terminals Located in Exhibit Area & Computer Terminal Rooms (Pay by the Hour) Use Library Programs or Write Own Programs | Lawrence Hall of Science |
| LEVEL II: General Public Access | Educational Computing Resource Center Applications Programs Written & Distributed On-Site Computer & Computer Specialist 1 or More Interactive Terminals in Exhibit Area with Repertory of Programs, Games, Information, etc. A Programming Class for Local Kids | The Boston Children's Museum Oregon Museum of Science & Ind. Boston Museum of Science Maryland Academy of Sciences Ontario Science Centre Pacific Science Center Franklin Institute Science Museum & Planetarium |
| LEVEL I: Special Public Access | School Field Trips and/or Special Classes, Workshops & Demonstrations for School Children, ages 6-18 One or More Computerized Exhibits Special Kids in the Basement | California Museum of Science & Ind. Chicago Museum of Science & Ind. Fernbank Science Center |
| GROUND LEVEL: Potential Public Access | Research, Data Processing, Cataloging and Record-keeping, Visitor Scheduling, and Controlling Equipment | A Majority of Science-Technology Centers and Museums |

Figure 1. A model of public access to computers through science-technology centers and museums.

ing to the management, and even a few who turn around and walk back out the front door, mumbling something about returning when the computer is up and running.")

Furthermore, while most exhibits in a science center are doing well to keep a visitor's attention for more than a minute or two (a few minutes in the case of live animals or manipulative puzzles), the computer generally must be programmed to encourage people to let others have a turn after five minutes or so, since visitors have been observed to spend 15-20 minutes or more playing games at computer terminals. And, while people tend to browse through exhibits and skim over or even ignore printed text (particularly if there is much of it), most computer terminals in museum exhibits communicate exclusively through text printed on CRT's or various typewriter-like terminals, and it would generally be correct to conclude that a museum visitor who is spending 15 minutes or more at a terminal must be reading—at least some of the time.

This scenario raises several interesting questions (as yet unanswered) regarding what people learn from playing games with

computer terminals. It also points out an important fact—namely, that people are fascinated by the power of interactive computing, particularly when it is presented in an informal, non-threatening, playful environment. Here, then, is evidence that the modern science-technology center is, indeed, an ideal institution for providing public access to computers, and in fact, that the computer potentially may be one of the most powerful educational tools in that institution. The realization of this fact and an institution's commitment to finding ways to put the power of computing in the hands of the public—to make computing accessible, understandable, and personal—truly distinguishes institutions at Levels II and III of our model from those at Level I. The Boston Children's Museum provides a good example of such realization and commitment. In a 1972 proposal to the National Science Foundation entitled 'Demystifying Computers,' Bill Mayhew⁹, coordinator of the museum's computer center, summarized the goals of his proposed project as follows:

'As technology grows in scale and capability, it is the responsibility of society to control it. To facilitate that control, each of us must learn how to deal with technology

sensibly, based on facts and not on misconceptions. One of the best ways for people to learn about things is to use them in a way that provides maximum flexibility in a creative atmosphere. In the case of computers, this means placing a variety of computing resources in the hands of the public, along with new and exciting ways to use them.' (p. i)

There are two main ways in which science centers and museums (including the Boston Children's Museum) have implemented Mayhew's goal: 1) by placing computer terminals and/or computer-controlled devices in public exhibit areas and 2) by offering computer classes to the public.

Computer Games for the Public. The most common approach currently used to give the public a hands-on introduction to computers is to place one or more interactive terminals in an exhibit area. Visitors are then challenged, implicitly or explicitly, to match wits with the computer by sitting at the terminal and playing a game of one sort or another. There are several generic categories of games which provide a repertory of programs popular with the public. While actual program names and implementations vary from one institution to another, one may expect to find some version of this standard repertory at any of these institutions.* These generic categories of programs are listed in Table 1 and examples of output from two such programs are shown in Figures 2a and 2b—compilations of computer games have recently been published and are listed in the bibliography^{1,11}.

Table 1. Categories and examples of interactive computer games and programs.

| CATEGORIES OF COMPUTER GAMES AND PROGRAMS | SOME POPULAR EXAMPLES |
|-------------------------------------------------------------------------------|------------------------------------|
| GAMBLING GAMES AND GAMES OF CHANCE | BLACKJACK, CRAPS, ROULETTE, CUPS |
| NUMBER GUESSING GAMES | GUESS, STARS, TRAP |
| WORD GAMES | WORDS, PASSWORD, HANGMAN |
| NIM-LIKE GAMES | 23-MATCHES, SIMNIM, CHOMP |
| STRATEGY BOARD GAMES | TIC-TAC-TOE, GOMOKU, QUBIC |
| HUNTING/COORDINATE SYSTEM GAMES | WUMPUS, HURKLE, BATTLESHIP |
| RULE GAMES AND GAMES OF LOGIC | IN&OUT, BAGELS, REVERSE |
| CONVERSATIONAL & INSTRUCTIONAL PROGRAMS | ELIZA, QUERY, MATHDRILL |
| ARTIFICIAL INTELLIGENCE; PROGRAMS THAT LEARN | ANIMALS, ENCYCLOPEDIA, HEX-A-PAWN |
| INTERGALACTIC WAR AND TRADING GAMES | STARTREK, SPACEWAR, STAR-TRADER |
| ECONOMIC, SCIENTIFIC, AND SOCIAL SIMULATIONS | STOCKS, KING, GENE I, POLICY, HOME |
| SPORTING GAMES AND OTHER SIMULATIONS | FOOTBALL, RACETRACK, LUNAR-LANDER |
| TELETYPEWRITER PICTURES AND PATTERNS | SNOOPY (et al.), AMAZE, POSTER |
| INFORMATION RETRIEVAL, ELECTRONIC BULLETIN BOARD, DATE, AND CALENDAR PROGRAMS | INFO, NEWS, BIRTHDATE |

In most science centers, a few games and programs such as those shown in Figure 2 (Table 1), are linked to a monitor program. This monitor program usually provides any or all of the following features:

*A further analysis and discussion of computer games and their social and educational ramifications is beyond the scope of this article. The interested reader is referred to a 1973 article that appeared in *Info-systems* entitled 'Computer Games People Play'⁵ and an article by Daniel L. Goldwater, Education Director of the Franklin Institute, entitled 'Games People Play—On Computers'⁷ which discusses the use of computer games in the Franklin Institute's exhibit area. Finally, Steward Brand's 1972 *Rolling Stone* article (also reprinted in *II Cybernetic Frontiers*) entitled 'Spacewar'² gives a good overview of current efforts towards and implications of bringing computers to the people.

- 1) a menu (i.e., a selection of programs from which to choose)
- 2) record keeping (e.g., how long which games were played and survey data such as age and sex of the players)
- 3) time keeping (i.e., insuring turn-around or turn-taking at the terminals)
- 4) 'public-proofing' of programs (i.e., parsing visitor input so as to recognize and effectively respond to (or ignore) null, extraneous, incomprehensible, random or obscene input)
- 5) 'escape-proofing' of programs (i.e., programmatically disabling the Escape or Break keys used to abort program runs. This function may also be accomplished through hardware modifications.

A series of articles on the Honeywell Computer Exhibit at the Boston Museum of Science¹² gives a good example of such a monitor program and also provides a multi-faceted view of that exhibit.

Computer-Controlled Devices for the Public. In addition to having public terminals, the Boston Children's Museum, OMSI, LHS, and other institutions have experimented over the years with exhibits that make various computer-controlled devices accessible to the public. Common examples include voice synthesizers, tone generators, graphics terminals and plotters, and electro-mechanical 'turtles.'

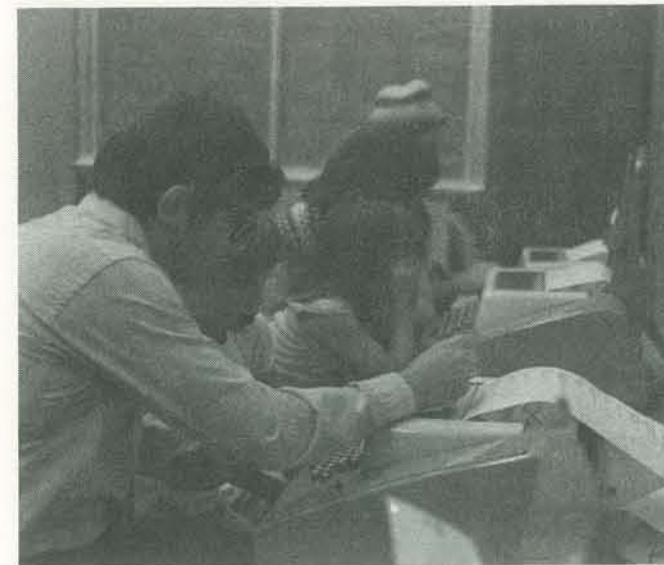


Figure 2a. Computing is a family activity at Lawrence Hall of Science.

```

XEQ-HOME.A001
WELCOME TO HOME - A HOME ENERGY USAGE PROGRAM

WHAT IS THE FLOOR AREA (ROUGHLY) OF YOUR HOME (SQ. FT.)? 1500
ARE YOU REASONABLY CERTAIN OF THIS FIGURE? YES
HOW MANY STORIES DOES YOUR HOME HAVE? 1
WHICH OF THE FOLLOWING BEST DESCRIBES THE INSULATION IN YOUR CEILING:
1) UNINSULATED
2) PARTIAL
3) FULL
1, 2 OR 3 ? 2
WHICH OF THE FOLLOWING BEST DESCRIBES THE INSULATION IN YOUR WALLS:
1) UNINSULATED
2) PARTIAL
3) FULL
1, 2 OR 3 ? 2
DO YOU HAVE A GAS OR ELECTRIC FURNACE? GAS
WHAT DO YOU SET YOUR THERMOSTAT AT DURING THE DAY? 66
WHAT DO YOU SET YOUR THERMOSTAT AT DURING THE NIGHT(0=OFF)? 0
HERE IS A SUMMARY OF WHAT YOUR ANNUAL HOME HEATING ENERGY SHOULD BE:

PART      $ENERGY      THERMS      $ COST/YEAR
CEILING   27            113.68      18.19
WALLS     19            79.9        12.78
WINDOWS  28            120.48      19.28
FLOORS    12            52.84       8.45
INFILTRATION 14            61.84       9.89
TOTAL     100           428.74      68.6

FOR MORE INFORMATION ABOUT INSULATING YOUR HOME, AND
OTHER ENERGY SAVING MEASURES, CONTACT YOUR LOCAL P.G.&E.
OFFICE. THANK YOU FOR VISITING THIS EXHIBIT.

TEAR OFF AND TAKE THIS PAPER WITH YOU.

```

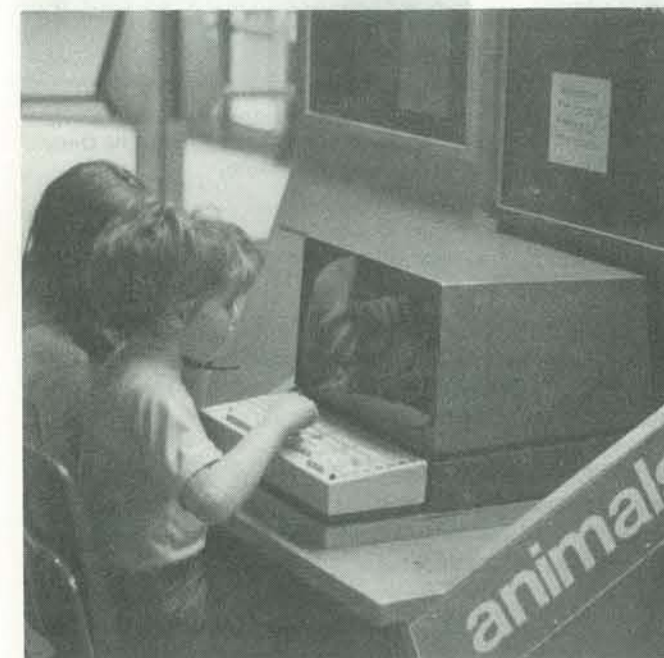


Figure 2b. Computer terminal in a public exhibit at Lawrence Hall of Science "learns" about animals.

```

XEQ-ANIMAL
THINK OF AN ANIMAL.
IF I GUESS WRONG, PLEASE HELP ME WITH A GOOD QUESTION.
ARE YOU THINKING OF AN ANIMAL?YES
DOES IT LIVE IN WATER?NO
DOES IT HAVE WINGS?NO
IS IT A DOMESTIC (HOUSE OR FARM) ANIMAL?NO
DOES IT HAVE 4 LEGS, NO MORE, NO LESS?YES
DOES IT HAVE A LONG NECK?NO
IS YOUR ANIMAL A TIGER
?NO
WHAT IS YOUR ANIMAL?PANDA
GIVE ME A QUESTION I CAN ASK TO TELL A TIGER FROM A PANDA.
THIS QUESTION MUST BE ANSWERED BY YES OR NO.
?IS IT BLACK AND WHITE?
FOR A PANDA YOUR ANSWER IS:(YES OR NO)?YES
THINK OF AN ANIMAL.
IF I GUESS WRONG, PLEASE HELP ME WITH A GOOD QUESTION.
ARE YOU THINKING OF AN ANIMAL?YES
DOES IT LIVE IN WATER?NO
DOES IT HAVE WINGS?NO
IS IT A DOMESTIC (HOUSE OR FARM) ANIMAL?NO
DOES IT HAVE 4 LEGS, NO MORE, NO LESS?YES
DOES IT HAVE A LONG NECK?NO
IS IT BLACK AND WHITE?YES
IS YOUR ANIMAL A PANDA
?NO
WHAT IS YOUR ANIMAL?ZEBRA
GIVE ME A QUESTION I CAN ASK TO TELL A PANDA FROM A ZEBRA.
THIS QUESTION MUST BE ANSWERED BY YES OR NO.
?DOES IT HAVE STRIPES?
FOR A ZEBRA YOUR ANSWER IS:(YES OR NO)?YES
THINK OF AN ANIMAL.
IF I GUESS WRONG, PLEASE HELP ME WITH A GOOD QUESTION.
ARE YOU THINKING OF AN ANIMAL?YES
DOES IT LIVE IN WATER?YES
DOES IT HAVE LEGS (OR FLIPPERS)?NO
DOES IT HAVE FINS?YES
DOES IT HAVE GILLS?NO
IS IT BIGGER THAN THE SIZE OF A CAR?NO
IS YOUR ANIMAL A DOLPHIN
?YES
HEY, GREAT! I GOT IT! THINK OF AN ANIMAL.
IF I GUESS WRONG, PLEASE HELP ME WITH A GOOD QUESTION.
ARE YOU THINKING OF AN ANIMAL?STOP

```

ment; thus, the visitor would type his/her comments at a terminal and ELIZA would talk back, as it were.) It should be noted that such an exhibit would have to be designed with great care to avoid mystifying rather than demystifying computer power.

Similarly, Lawrence Hall of Science has an exhibit using a computer-driven tone generator; i.e., a simple electronic device for producing tones of various frequencies under computer control. The tone box, as it is called, is connected through a terminal in the exhibit. Using a suitable musical notation (e.g., the keys 'C,D, E, F, G, A, B' on the terminal keyboard represent the notes DO, RE, MI, FA, SO, LA, TI of the musical scale, the space bar produces a Rest, etc.), a

visitor may compose a tune that the computer will store and play back on command. There are also options for playback with variations in tempo and pitch. LHS maintains a library of tunes 'transcribed for computer tone box' by visitors and 'special kids.' These tunes may be requested by name at the terminal. Popular tunes include Scott Joplin's 'The Entertainer,' a Bach Prelude, 'Fight for CAL,' the theme from *Love Story* and many more.

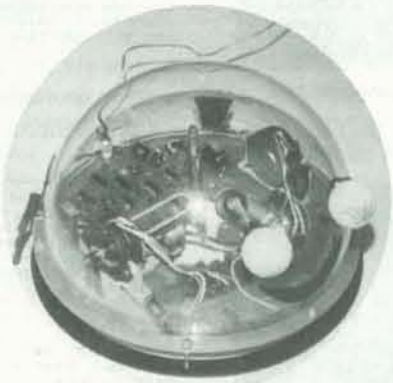
Graphics is another excellent way to introduce the power of computers to the public. Several different types of graphics devices have been used successfully in the museum environment. The least sophisticated and least expensive of these is the cursor-addressable, character-generating CRT terminal



Composing a tune on Lawrence Hall's computer-driven tone generator.



Computerized space war. Button boxes give players control of thrust, orientation and torpedo fire.



A computer-driven turtle.



Calendar program displays the day of the week for any date in BIG LETTERS at the Ontario Science Center in Toronto.

on which it is possible to selectively change one part of the screen while leaving the remaining display unaltered. Such CRT's allow a game such as Hangman to be programmed so that the 'man' does not roll off the screen and have to be completely re-drawn each time a letter is incorrectly guessed. Rather, arms, legs, trunk, etc., are simply added with each missed letter until either the word has been guessed or the man is completely changed.

Fully refreshable vector graphics CRT's or plasma displays with graphic input capabilities would be most desirable in the museum environment since they provide a quantum leap in power and flexibility. Such devices have been out of the range of very limited museum exhibit budgets, but continuing advances and declining prices in microcomputer technology are rapidly changing this situation. One option currently available to museums is the Tektronix storage tube graphics terminal, such as those being used at OMSI. These terminals have the advantage of being durable and low cost while providing both character and vector graphics with an option for graphic input. Another contender for the museum exhibit market is the packaged micro computer system. Such machines as the Commodore PETTM, Radio Shack TRS-80TM, Exidy SorcererTM and particularly the Apple IITM and CompuColor IITM with color graphics would certainly provide a quantum leap in capability from the status quo.

Another type of device that has been used effectively in some science centers to introduce computers to young children is

Seymour Papert's robot 'turtle'. (This use of the name turtle probably derives from the electro-mechanical tortoise built by physiologist W. Grey Walter in the late 1940's to study cybernetics.) To quote Sema Marks³:

'A turtle is a computer-driven supertoy. It can move forward or backward and can rotate around its center. A pen is attached to its center, and when the pen is down, the turtle draws a line wherever it goes (p. 103).'

Turtles are commercially available, although some museums have built their own from Meccano sets or from spare parts. The movement of a turtle may also be simulated on a graphics plotter or graphics terminal.

Using a programming language called LOGO, children learn to control the motion of the turtle, thus expanding their knowledge of what Papert calls 'turtle geometry.' Papert¹⁰ feels that the experience of a program-controlled device can be used to 'give children, to quite an unprecedented degree, a sense of power of ideas in general, of science in particular and especially of mathematics (p.8).' By learning to control the turtle, children master concepts such as planning, debugging, modular structure, modeling and so on, thus giving them a mathematical experience more like an engineer's than like a bookkeeper's. One may simply extend these ideas to the museum environment analogizing that program-controlled devices can also give this sense of power to adults as well as children.

Table 2. Computer classes offered at various science centers.

| INSTITUTION | COURSE OFFERED | ENROLLMENT INFORMATION | COURSE DESCRIPTION |
|--------------------------------------------------------|---------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| The Franklin Institute, Philadelphia | Computer Programming | Grades 8-10 6 Saturdays (9:30 to 11:30 plus laboratory time) Members \$32, Non-members \$35 | Workshop emphasizes computer language. Once familiar with the APL terminal and language, students will understand how computer performs rapid calculations and stores information. They will play computer games and practice writing and testing their own programs. |
| Oregon Museum of Science and Industry, Portland | Pascal I—Computer Class for Beginners | Age 12 and up 6 Tuesdays (7 to 9 p.m. plus one-hour computer use) Enrollment limited to 12 students Members \$25, Non-members \$28 | Pascal computer language, developed in Europe for educational use, will be taught to those who have some experience in math or math puzzles and ability to solve $5x + 17 = 3$ for the value of x. Students will use PDP-11/45 computer to test programs they have written. |
| | Pascal II—Intermediate Computer Programming | 6 Wednesdays (6:30 to 9:30 p.m.) Members \$30, Non-members \$34 | Continues Pascal I class, which is a course prerequisite. |
| | Pascal III—Advanced Computer Programming | 6 Thursdays (6:30 to 9:30 p.m.) Members \$30, Non-Members \$34 | Continues Pascal II class, which is a course prerequisite. |
| California Museum of Science and Industry, Los Angeles | Computers | Grades 5-8 4 Saturdays \$15, plus \$2 Lab Fee | Gather data on how the computer thinks, learn how to work the apparatus, and write programs using its special language. Challenge the computer to games that test your wits. Field trip to a computer research center included in the last session. |
| Fernbank Science Center, Atlanta | Introduction to Computer Programming | Four levels offered Student-adult, grades 7-12 Teacher in-service Elementary or High School Family 5 evenings (7:30 to 9:30) Registration Fee \$6 | Introduces students to computer programming in APL and gives them enough information to write complete programs for problem solving. At end of course, student will be able to continue with more advanced programming features on his own initiative. |
| Lawrence Hall of Science, Berkeley | Creative Play with the Computer | Age 8-11, grades 3 to 6 8 afternoons (4 to 5:30) Enrollment limited to 16 Members \$42, Non-members \$50 | Explore computer as an artistic, expressive, and recreational medium. Learn to use teletypewriter and electronics graphics plotter. Other activities may include use of a computer-controlled musical tone box, cathode ray tube terminal, and a graphics display screen. |
| | Beyond Creative Play | Age 10-13, grades 5 to 8 8 afternoons (4 to 5:30) Enrollment limited to 16 Members \$42, Non-members \$50 | Students learn problem-solving techniques that prepare them for beginning programming. Computer graphics and games will be included, but activities are directed toward learning how to write simple programs. |
| | Computer Science Seminar I | Age 13—adult, grades 8 to adult 8 afternoons (4 to 6) Enrollment limited to 12 Members \$42, Non-members \$50 | Explore advanced topics in computer science on an interactive, small-group basis. Depending on experience of participants, activities may include machine organization, advanced programming languages other than Basic, and writing programs for specific applications. |
| | Pilot Your Own Computer | Age 10-16, grades 5 to 11 8 afternoons (4 to 5:30) Enrollment limited to 15 Members \$42, Non-members \$50 | Learn to program in Pilot, a non-math computer program. Materials and instructional guidance help students to progress at their own pace, learning concepts that provide a good foundation for programming in Basic and other computer languages. |
| | Computer Programming in Basic | Age 10-18, grades 5 to 12 8 afternoons (4 to 5:30) Enrollment limited to 15 Members \$42, Non-members \$50 | Class will be divided into small groups on basis of previous experience. Individual instruction within groups. New topics in Basic programming will be introduced and explored. With guidance of instructor, students will develop their own computer programs. |
| | Computer Exploration for Adults | 8 evenings (7 to 9) Enrollment limited to 12 Members \$32, Non-members \$40 | Not a programming class. Programming languages, computer systems, and computer hardware will be explained in understandable terms. Explore ways which computer affects daily life in terms of business, education, art, science, privacy, and law enforcement. |

Computer Classes. Classes at science centers provide another way of increasing public awareness about computers, and in more depth than might usually be achieved through exhibits.

Most science centers offer afternoon, evening, or weekend classes in a variety of science-related subject areas to children and adults in the local community. The classes are most often

taught by members of the institution's staff having particular interest or expertise and they are supported, in part, by tuition fees (often with discounts for members of the institution). They are usually informal, workshop-style, discovery-oriented classes, offered for enrichment rather than for credit.

Table 2 shows sample descriptions of computer courses offered at various institutions. Notice that most of the institutions listed offer only one course aimed mainly at students ages 11 and up. The descriptions generally emphasize learning basic computer concepts, playing games, and especially learning to program in APL, BASIC, PILOT or even PASCAL at OMSI. With the exception of Lawrence Hall's extensive offerings, these classes would best be categorized as another form of limited or special access in which a relatively small fraction of the general public has an opportunity to learn about computers in some depth and to learn a programming language. These courses are included in the description of Level II because they are available to the *general public* and not just to special school groups. Furthermore, a real need is for many more courses such as Lawrence Hall's 'Creative Play with a Computer' (for 8-12 year-old children) and 'Computer Exploration for Adults.' These courses concentrate on what computers are, how they work, their uses and abuses, and how they affect the daily lives of people rather than concentrating on learning programming per se.

Three Educational Computing Resource Centers. Certain Level II institutions have made a sizable investment in computer resources while others with limited resources have at least attempted to make those resources available to the public. Three institutions, the Boston Children's Museum, OMSI, and LHS operate on-site, mini-computer, time sharing systems, thus assuming a role of entrepreneurship not possible in most other institutions. All three of these centers have at least one staff member specifically dedicated to computer-related activities, and all three work closely with local schools. Indeed, one of the main functions of these institutions is serving as an educational computing resource center for local schools.

In addition, LHS is engaged in providing an educational time-sharing service to local schools. The LHS computer system presently reaches more than 40 schools and education institutions in Northern California including a Montessori school, elementary, junior and senior high schools (both public and private), and some departments on the Berkeley and UCLA campuses of the University of California.

OMSI and the Children's Museum have more limited facilities for remote time-sharing, and thus they have concentrated their efforts on software development. Marketing of system software developed at OMSI for the PDP11/45 has been one of the sources of revenue helping to support the OMSI computer center. All three institutions work with teachers in local schools to provide both computer programs and computer-based materials for use in the classroom.

To summarize Level II, General Public Access: Science centers and museums at this level are making computer power openly accessible to the general public by placing computer terminals and computer-controlled devices in public exhibit areas and,

to a small extent, by offering classes. The computer exhibits at Level II institutions are of an informal rather than formal nature. At Level I institutions, the power of the computer is formally demonstrated to an 'audience' several times a day by a staff member of the institution; the terminal or device is not otherwise visible or available to the public. At Level II institutions, people may use the computer terminal (device) at their own pace and in their own manner. The exhibit is always accessible (at least when it is working). Classes, if offered at Level II institutions are open to the general public, i.e., they are not just for school groups, and they are for adults as well as children.

In the final part of this article, which will appear in the next issue of *People's Computers*, these distinctions will be amplified and extended. This part of the article will focus on the Lawrence Hall of Science as an example of an institution that has gone beyond providing public access to assuming an active role in public computer education. I will also discuss some characteristics of our 'ideal' public access center and will conclude with some speculations on the role of public access centers in the future. □

References

1. Ahl, David H. *101 BASIC Computer Games*. Digital Equipment Corporation, Maynard, Mass. 01754, 1973.
2. Brand, Stewart. *II Cybernetic Frontiers*. New York: Random House/Bookworks, 1974, pp. 39-90. Originally published in *The Rolling Stone*, December 7, 1972, pp. 50-58.
3. Brown, Les and Sema Marks. *Electric Media*. New York: Harcourt, Brace and Jovanovich, 1974, pp. 96-158.
4. 'Bursting at the Museums.' *Computer Decisions*, vol. 7, no. 3, March 1975, pp. 44-45.
5. 'Computer Games People Play.' *Infosystems*, vol. 20, no. 10, October 1973, pp. 26-29. Also in: Van Tassel, Dennie. *The Complete Computer*. Palo Alto: SRA, Inc., 1976, pp. 50-54.
6. Davis, Kay and Lee Kimche, ed. *Survey of Education Programs at Science-Technology Centers*. ASTC, 2100 Pennsylvania Avenue, N.W., Washington, D.C. 20037, 1976. \$4.50 incl. postage and handling.
7. Goldwater, Daniel L. 'Games People Play — On Computers.' *Museum News*, vol. 51, no. 8, April 1973, pp. 30-34.
8. Kemeny, John G. *Man and the Computer*. New York: Charles Scribner's Sons, 1972.
9. Mayhew, Bill. 'Demystifying Computers: A Proposal to the National Science Foundation to Investigate New Ways for People to Learn Through the Use of Computers.' The Children's Museum, The Jamaica Way, Boston, Mass. 02130, October 1972.
10. Papert, Seymour. *Uses of Technology to Enhance Education*. LOGO Memo no. 8, Massachusetts Institute of Technology, Artificial Intelligence Laboratory, 545 Technology Square, Cambridge, Mass. 02139, 1973.
11. People's Computer Company. *What To Do After You Hit Return* (or PCC's First Book of Computer Games). People's Computer Company, P.O. Box 310, Menlo Park, CA 94025, 1975.
12. Spangle, C. W. et al. 'The Honeywell Computer Exhibit at the Boston Museum of Science.' *The Honeywell Computer Journal*, vol. 8, no. 1, 1974, pp. 7-16.

Continued from page 7.

amples given above, by using INKEY\$ you can get input and keep complete control over what appears on the screen, plus the standard input routine does not crash when given too much, too little, or incorrect data. And with CLOAD? you can make sure that a good copy of a program has been saved without destroying the copy in the machine. Some 'picky' details also have been handled well. For example, in formatted print statements (i.e., PRINT USING) values that are too large for their field specifications are printed in their entirety with a special flag; they are not truncated or thrown away as with so many systems.

There are still a few rough spots, although they may or may not be apparent to a novice consumer. For example, one-character file names for tape are somewhat limiting; the susceptibility to volume variations of the tape systems (i.e., having to load a tape at four different volumes before you find one that works) can get tedious in a hurry; and the fact that the system is still not particularly fast—although Level II is about 40% faster than Level I, it is still in the slower half of the field according to the Feldman-Rugg Benchmarks (*Kilobaud* No. 10)—is sometimes noticeably apparent.

Documentation. Radio Shack has done a good job with software documentation. Hardware documentation is mostly non-existent, but most people don't seem to be bothered by that. Both the Level I and II manuals are clear and accurate. Most of the Level I manual is 'old hat' to anyone who already knows BASIC, but that is to be expected. The Level II manual assumes prior knowledge of BASIC and suggests that users who don't know BASIC go through the Level I manual first.

Overall Opinion. I would recommend the Level II machine but not the Level I. I just cannot imagine anyone remaining satisfied with the limitations of a Level I machine. The Level II is a different story; while it is not 'snazzy' and it won't appeal to heavy hardware and software 'freaks', that is not its intended market. It is a reasonably priced, readily expandable machine, with a good BASIC and clear documentation. So, in the final analysis, the arguments for or against the TRS-80 remain what most arguments are—a matter of taste. —Eryk Vershen □

REVIEWS

ILLUSTRATING BASIC
(A Simple Programming Language)
By Donald Alcock
Cambridge University Press, 1977
134 pp. \$4.00 (approx)

Computer language manuals that are both readable and useful are hard to find, but Donald Alcock's soft-cover BASIC manual belongs to that rare breed. As a programmer of scientific and educational applications software, I have often been frustrated by obtuse manuals, especially those supplied by mainframe manufacturers. *Illustrating BASIC* is far above the usual gibberish.

Its organization into nine cohesive chapters makes the book easy to use. A novice programmer, moving through the chapters in sequence, would be writing programs after Chapter 1, 'Components of the Language.'

'Heavy-duty' subjects such as arrays and matrices have their own step-by-step explanatory chapters, and even simple GO TO instructions get the kind of treatment which captures the varied applications of the command.

Alcock also discusses the nuts and bolts of signing on, files, and other practical topics often left for oral presentation by an instructor. Transportability, or the ability to move a program from one machine to another without editing, is a concern which receives attention throughout the book. As most programmers know, implementing a program on a new machine can be a major effort.

The ninth chapter, 'Syntax', could be an appendix. This chapter simply gives the Backus-Naur (standardized format) syntactic rules of the language, with the unusual twist that examples and comments follow the potentially confusing rules. Some authors get so carried away with the elegance of simplicity in their syntactic rule presentations that the reader ends up lost or bored; not so with Alcock.

Equally important to the message of the manual is the medium. Alcock's book

is both stylish and clever. It is a spiral-bound pocketbook printed entirely in hand-lettered capitals. The text is liberally sprinkled with illustrations—bold arrows and comic book 'flashes.' A talking beetle represents potential coding bugs. This approachable, attention-getting format reinforces the idea that learning to program can be simple, non-threatening, and even fun.

The manual is written for computer programming beginners. I would estimate that it is appropriate for junior high school age and above. With its clever graphics and instructional clarity, it should be especially appealing to students.

Reviewed by Chuck Dunbar

THE C PROGRAMMING LANGUAGE
By Brian W Kernighan & Dennis M Ritchie
Prentice-Hall, \$10.95

There has been a lot of talk recently about the C programming language. Unfortunately, however, there has not been much written about it except a short reference manual. This book changes that.

While not suitable for the beginning programmer, it should be intelligible to anyone who has programmed in at least one language and who is familiar with basic concepts like variables, assignments, loops, and functions. The presentation is somewhat biased towards the current PDP-11 UNIX™ version of C, but the authors do attempt to point out differences in earlier versions and versions on other machines.

Explanations are based on examples, most of which are complete programs. The authors also point out a number of mildly cryptic idioms that are commonly used in C programs.

The book includes the current C reference manual as an appendix.

Reviewed by Eryk Vershen

FORTMAN

BY LEE SCHNEIDER &
TODD VOROS
VOL. 3 EPISODE 4

In our last episode, the tiny chip carrier containing the PROMs into which Fortman had placed himself is discovered floating in the data channel by members of the Underground Resistance Movement... and, with the aid of a Relocatable Loader, Our Hero is at last reloaded back into normal execution.

But wait: it is not so normal after all! For when Billy Basic tried to smuggle the PROMs past the border into Microprocessor Land, an alert guard exposed them to ultraviolet light... and now, although he retains his super-software-powers, part of his memory has been erased... and Fortman Man cannot remember who he is or why he is there!

Linea, the dedicated commander, leading this band of resistance in their fight to free the Land of the Little People from the oppression of the evil Glitchmaster, attempts to question him, with little result... and then, in the midst of the interrogation, the camp is attacked!

As two columns of Random Numbers descend upon the secret rebel base, Fortman Man, for reasons he cannot explain, feels compelled to compile, and with unheard-of speed, he single-handedly defeats the entire attacking force.

Although there is no data available on the source of this new code, Linea nevertheless accepts Our Hero immediately into the ranks of the Resistance, and with this powerful new ally, makes plans to attack the stronghold of the Glitchmaster Himself!!!

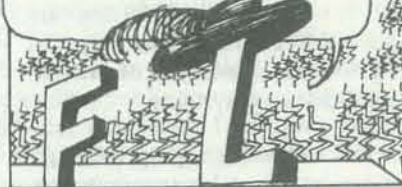
In serial format they advance out across the wide data fields, being careful, as always, not to leave behind any altered variables or broken branch statements that their enemies could use to trace their flow.

Orders are issued, the routines are assembled... and, early on the rising edge of the new cycle, Linea addresses her troops...

You of the resistance... listen to me! Our beloved land has been suppressed under the tyrannical noise of the Glitchmaster long enough! The time has come, comrades, when we of the resistance shall pull down the current regime, and restore clean, decent signals once again to Microprocessor Land!

I have sent word to General Wirewound, and his forces shall march to meet ours... and, with our new ally who stands here beside me, we will win this battle!

All right; pick up your resistor packs, and let's march!



The journey goes on... then suddenly they are halted, as in their execution path they stumble upon a set of strange markings at ground level...

What is it?

I'm not certain, but they look like fresh disc tracks! I'd say a disc transport has been moved through this area recently!

But... it can't be ours... and according to my data the Glitchmaster doesn't have any discs in this sector!

Perhaps, but I suggest we follow them and find out for sure!

With the addressing task completed, the resistance fighters form themselves rapidly into parallel ladder networks and climb slowly out of the hidden valley, pulling up their supplies behind them...

On Linea's suggestion they quietly trace down the path in search of its source code, and shortly thereafter, the object they seek is located, cleverly concealed in a cleared sector...

All right then, let's give them a little surprise visit!

You, stranger, take one decade of resistance troops and branch around to their positive side...

And you take another megohm and some underground supplies and circle around to their negative side!

When I initiate the signal, both of you attack... you'll meet in the middle and drive them right into ground level!

Yes, commander!

The resistance movement is rapidly completed, all components are properly positioned... the signal is given, and the battle is on!

Aha! Just as I suspected... a secret data base! And with a full set of off-line discs...

That's simple, the rest of the squadron is probably searching the data fields for us!

Doesn't seem to be anyone here but a few guard bits... I wonder why?

In her usual efficient manner, Linea turns and rapidly issues instructions.



The guards are swiftly overflowed by the well-timed assault... and now, at Linea's order, the resistance fighters and their supplies are quickly loaded onto the captured discs...

This was indeed a fortunate random occurrence!

With these discs we can short-circuit over Core Plains, attack the Glitchmaster and reload the old regime in multi-record time!



Careful with those supplies! Be sure everything is loaded correctly... there isn't time to do a Verify!

All right... activate those drives and bring them up to speed!

The drives are switched into positive mode, and, upon removing the grounding straps, rise swiftly above ground potential...

They begin to pass over the plains, and as predicted, the magnetic radiation from the cores begins to cause slight turbulence as the discs pass over them...

But then there is a sudden, drastic increase in the oscillations...

W... W... What's... h... happening?

Linea... the other discs are all logging the same errors... this is impossible!!!

Hey... we're losing disc speed! What happened???

I don't know! T... The noise shouldn't be t... this severe!

Our drive has been neutralized!

Hang on!!!!

This might be a bit of a rough ride, but the Glitchmaster would never expect us to attack from this direction... and therefore, we will!

I only hope General Wirewound gets here in time!

Unexplainably de-selected and out of control mode, the discs begin going down with alarming rapidity, until...

Our... our data transfer was abnormally terminated... but why?

GASP! It's... it's...

The LOCKOUT MONSTER!!!!

I'm not sure... but...

Hey! Look out!



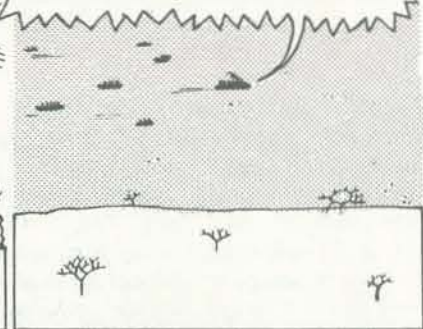
Slowly, the members of the resistance force stagger out from beneath the wreckage of their crashed discs...

Swiftly they travel over the data fields... and as they approach the leading edge of the core plains, Linea broadcasts a final warning...

Attention all discs!

The core plains are ahead... prepare for possible magnetic force turbulence! Check your tolerances for proper noise immunity settings!

Access time to target is now 3.7 milliseconds!



And then, just as suddenly...



Swiftly they turn in response to Fortman Man's shout of warning... and as they do, a great dark shadow falls over the field...

Is this the END for our heroes? Is the resistance movement fated to be terminated forever, grounded in the core plains? Is the Glitchmaster ever to be defeated? Will Billy Basic, or for that matter, Our Hero, ever see the high-level structures of 360 City again? Or will the Lockout Monster disconnect them forever?

Tune in again next issue... same timing track, same sector!

COMPUTERIZING MUSEUM EXHIBITS

THE BEGINNINGS OF THE ESSENTIAL CONVERSATION

BY BRUCE BURDICK

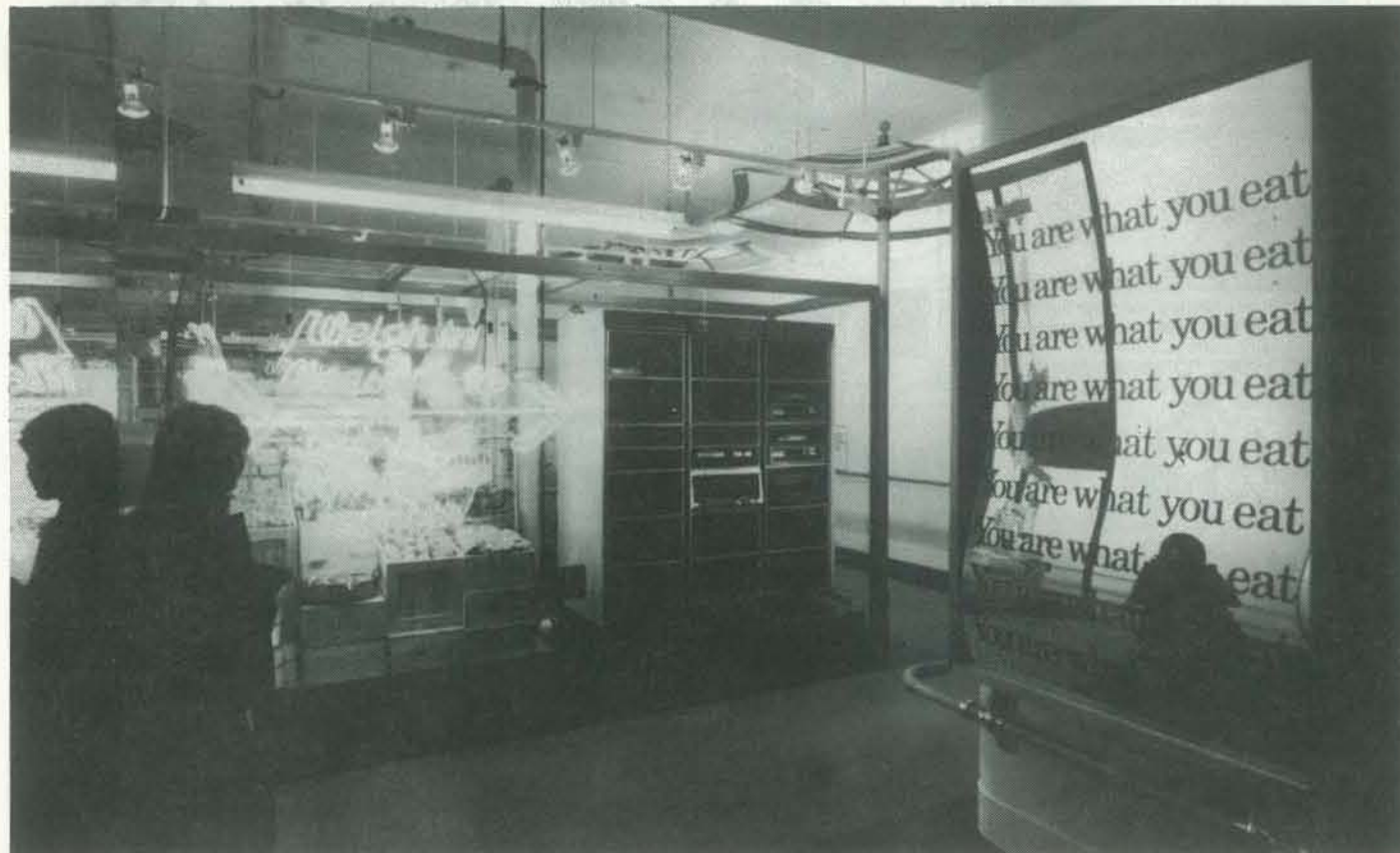
Bruce Burdick is the president of the Burdick Group located in San Francisco. The Burdick Group is engaged in design related to man, his environment, products and communications. One of their primary interests is designing exhibitions for science museums and fairs. This article explores the philosophy and implementation of a computer-based nutrition exhibit which was commissioned by Swift and Co/Esmark and opened at the Chicago Museum of Science and Industry in 1976.

Building on the success of this exhibit, the Burdick Group is now exploring ways to expand the use of computers in designing interactive museum exhibits. They are currently developing an exhibit on economics and banking using color graphics—also to be housed in the Chicago Museum—and they are developing a travelling exhibition on creativity which will be based around visitor interactions with a computer. — BK

How would you like to go to a science center or a museum, have it 'recognize' you, remember pertinent facts about you, and be able to converse with you, both about the things that you know and would like to know?

'To have a museum be responsive in real time to the individual's curiosities. . . requires the museum and the visitor to be able to enter into a conversation.'

If that were possible, museums would no longer be deaf and dumb to their visitors. Such an innovation has taken place in 1976 with the opening of the Swift/Esmark exhibit on nutrition at the Museum of Science & Industry in Chicago.



Entryway into the Swift/Esmark Nutrition Exhibit at the Chicago Museum of Science & Industry. The neon sign, a little unclear in the photo, says "weigh in, measure up, get your computer number."

'We quickly saw that the computer is more than an information machine. It is, in fact, a different form of media, and, as such, can and should interact with the things around it . . .'

If the social value of museums is to be realized—that is as resources to their communities—the initiation of an essential conversation between the facility and the visitor is the inevitable first step. Museums have long been dependent upon people to have with them a large bag of information for the understanding of the museums' subjects. This is because the activity of the museum has been essentially the act of display. Yet, though our eyes are a rich resource as receivers and inquirers, to utilize them alone is to separate us from other ways of knowing things; that is, through inquiry participation and direct experience.

It is not that museums have not tried to move past the act of display; they have—by the use of sound tapes, graphics, lectures and walking tours. We have all experienced these types of media and their limitations. They usually are selectively blind to who I am, what my intellectual background is, how much time I have for my visit and what my curiosities are. You and I have been *missed* by this type of available information.

These methods of communication face and solve their communication problems the way that television does—by writing something that must appeal to all ages and educational levels. The end result of this approach for museums and television is to reduce what is communicated to a mythical norm (a norm that cannot be interacted with and that no one fits).



Computing nutritional values: a video display inside the exhibit case and a numerical pad outside allow visitors to interact with the exhibit.

Museums lack utility to the individual because individuals are constantly establishing and re-establishing what *they* want, when *they* want it—allowing their curiosities to establish their agendas. Such shifting of personal agendas frequently is not reflected by the institution. To have a museum be responsive in real time to the individual's curiosities, to meet them where they are now, requires the museum and the visitor to be able to enter into a conversation.

What can be done to create this two-way conversation, regardless of visitor's age, background and education, so that the museum can be a true resource to the community? This was a subject studied by our Group over two years ago. To do this, we needed a vehicle whereby inquiry and conversation could take place and we became intrigued with the idea that the computer had these capabilities. We fantasized a computer program that could allow the museum visitor to enter key pieces of information about himself that the computer would retain and thereby 'know' the visitor. Then, as the visitor moved about the museum and asked for information, the computer would respond specifically to that individual's background and informational needs.

Visitor #213 is 42 years old, an architect, from Atlanta, Georgia. The computer is replying to a question on Thomas Jefferson:

VISITOR #213: JEFFERSON WAS YOUR AGE AND ALSO AN ARCHITECT WHEN HE WENT TO FRANCE.

WOULD YOU LIKE TO:

- SEE HIS NOTES ON THE ARCHITECTURE OF FRANCE? PUSH #1.
- VISIT ARCHITECTURE INFLUENCED BY HIM IN THIS CITY? PUSH #2.
- OBTAIN A LISTING OF ALL EXHIBITS AND TOPICS ON JEFFERSON AVAILABLE TO YOU TODAY? PUSH #3.

We envisioned a program that would be expansive in nature. By expansive, I refer to that experience that we have all had when looking for a particular book on a library shelf, only to find that the books on either side were of equal interest. A computer can duplicate this type of experience. While handling your specific information request, it can also enlarge upon those subjects related to it.

...we decided to build an abstracted market within the museum, whereby with the use of the computer, the visitor could shop for nutritional information.'

So our fantasy program would relate to your immediate needs and be expansive in scope.

Visitor #344 is 13 years old, pursuing a science project. The computer has been programmed for her class's visitation this week:

VISITOR #344: SPECIFIC TO YOUR 8TH GRADE SCIENCE CLASS ARE OTHER EXHIBITS HERE THAT YOU MIGHT ENJOY:

- POLLUTION AND THE ENVIRONMENT
- AN ECOLOGICAL SYSTEM

WOULD YOU LIKE TO EXPLORE THESE?
YES: PRESS #1. NO: PRESS #2.
ALTERNATES TO THESE: PRESS #3.

We quickly saw that the computer is more than an information machine. It is, in fact, a different form of media and, as such, can and should interact with the things around it rather than be an end result in itself. As an example, a computer can answer questions about objects or events in its immediate vicinity; it can refer to an object adjacent to it (or elsewhere in the museum); and it can refer to a book for greater knowledge. At times it might recommend that the visitor leave the museum now and experience both the question and answer in the home or in the city.

Visitor #812 is visiting a science center in San Francisco:

VISITOR #812: YOUR ANSWER IS CORRECT. YOU MIGHT LIKE TO EXPERIENCE BERNOULLI'S PRINCIPLE BY WALKING DOWN MAIDEN LANE. WATCH HOW THE WIND VELOCITY CHANGES AS YOU ENTER THE LANE.

Somewhere in our thought process, the opportunity came along which enabled us to move out of the realm of planning into the integration of our evolving ideas into an educational situation.

Swift & Co/Esmark asked us to examine their existing exhibit on nutrition at the Museum of Science & Industry in Chicago and to develop a new permanent program for its replacement. Intriguing to us was the fact that Swift, in its food research laboratory, had a computer well-stocked with scientific data



Comparative shopping in a museum market. A CRT among the realistic-looking foods displays the menu and nutritional values.

on the nutritional value of almost every food consumed in the United States. Here was an almost inexhaustible data library on nourishment.

Nutrition is a terrible subject. After all, each of us is an 'expert.' We are involved in the nutritional ritual four or five times a day and our mothers and teachers have ground into our heads what is good for our bodies. All of this has locked certain food biases in us, resulting in a lack of useable nutritional knowledge and an abundance of food fallacies.

Surrounded by this abyss of misunderstanding, our Group started conceptual work to develop an educational situation. We decided that it was within the marketplace that the final

'Now, as a visitor to a specific area of the exhibit, you could ask nutritional questions and receive answers specific to your physiology.'

nutritional battle was won, lost or compromised, because once the food is purchased, it is more than halfway on its journey to our stomachs.

For this reason and the fact that markets are partially arranged into food groupings—a concept used in nutritional education—we decided to build an abstracted market within the museum, whereby with the use of the computer, the visitor could shop for nutritional information.

A video screen linked up to the computer would greet you at the entrance to the market:

WELCOME

YOU CAN SHOP FOR NUTRITIONAL INFORMATION WITHIN THIS UNIQUE MARKET BY USING YOUR COMPUTER NUMBER.

PLEASE ANSWER THE FOLLOWING QUESTIONS:
(AGE, WEIGHT, HEIGHT, SEX)

You would enter this information, utilizing a simple numerical key pad which we designed. The computer would then remember you individually by issuing a personal number. This would be entered each time you had a nutritional question, with the resulting reply tailored to your personal nutritional needs.

HELLO VISITOR #111: YOU ARE A FEMALE, AGE 32, WEIGHT 109 LBS., HEIGHT 5'-4". I WILL REMEMBER YOU FOR 2 HOURS.

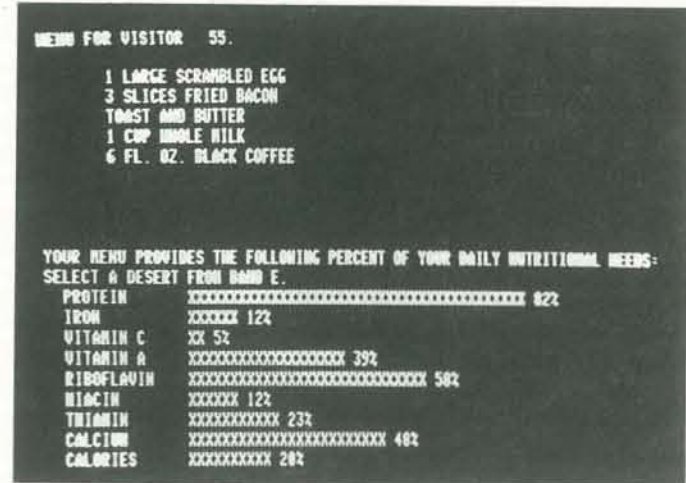
PLEASE USE YOUR COMPUTER NUMBER 111 WHEN ASKING QUESTIONS.

For almost all of us, that reply would be the first time that a museum exhibit 'knew' anything about us. Now, as a visitor to a specific area of the exhibit, you could ask nutritional questions and receive answers specific to your physiology.

VISITOR #111: THE ORANGE THAT YOU SELECTED WOULD PROVIDE THE FOLLOWING PERCENTAGE OF YOUR DAILY NUTRITIONAL NEEDS:

You could construct entire meals and test the results, changing entries when desired.

VISITOR #111: YOUR SELECTED MENU PROVIDES THE FOLLOWING PERCENTAGES OF YOUR DAILY NUTRITIONAL NEEDS:



Close-up of visitor #55's selected breakfast with an analysis of its nutritional value.

And you could receive an evaluation of your selection.

VISITOR #111: PLEASE NOTE THAT THIS MEAL DOES NOT PROVIDE YOU WITH 1/3RD OF YOUR DAILY NUTRITIONAL NEEDS.

What we have developed for the Museum of Science & Industry is a first-generation type of response possible from a computer. In second-generation programs, the computer will possess the capacity to know the visitor in greater depth. These programs will be responsive to the educational levels of the visitor, his particular areas of interest, his reasons for visiting the museum and the time available for this visit.

We are currently developing plans for a museum in which most of the information conveyed will appear through the use of video systems tied to a computer. In this museum, the visitor will be remembered by the computer for an all-day visit. We are also envisioning museums in which a membership will qualify the holder to be permanently 'stored' within the computer so that each visit will take up where the previous one left off.

Computers are obviously contributing more and more to our society: in industry, in universities, and in government. At some point, the museum and its visitors will be freed from their mutual blindness to one another. Museums will be true resources to their communities—responsive marketplaces for ideas, events, and knowledge. And it will all start with the push of a button. □

DECIMALS

in tiny BASIC

BY MILAN D. CHEPKO

Milan Chepko is a doctor living in Thief River Falls, MN who spends his spare time writing Tiny BASIC programs on his home-brewed micro. In the past two issues, Milan has given us Tiny Blackjack (see Letters section of this issue) and Tiny Concentration. Milan recently purchased a TRS-80 (anyone interested in buying an S-100 8080A system?), so we now have another good source for TRS-80 programs as well as Tiny BASIC. — BK

For the last 8 months, I've been running my 8080-based micro with Denver Tiny Basic (DDJ March 76). My choice of TB over a full Basic was dictated by convenience and expense: I couldn't find an inexpensive, easily-adaptable Basic, and by using TB I have more of my 8K RAM available for programs, making additional memory boards unnecessary. Denver TB has most of the features needed for games and other activities, but occasionally I miss the decimal capability of full Basic.

This limitation can be overcome by converting decimal numbers to integer numbers at the time they are input, and then converting back to a decimal form before printing. The following is a very simple method of accomplishing this with only a few steps, restricted only by the integer limit of TB (± 32767).

First, the number is input as two separate integer variables, with the decimal forming the point of separation. For example, to input the value 12.3 we would set A=12 and B=3. These are then combined into a single integer by multiplying the 12 by 10 and adding the 3, giving 123, which can be manipulated as a single variable by TB. Denver TB allows the use of a decimal point, comma, or space to separate two or more variables being entered on the same line, so the response can be 12.3 or 12,3 or 12 3. This input routine can be summarized as follows:

```
IN A, B      respond with 12.3
A=A*10      increases A from 12 to 120
X=A+B       adds A and B for total of 123
              and stores it in X for later use
```

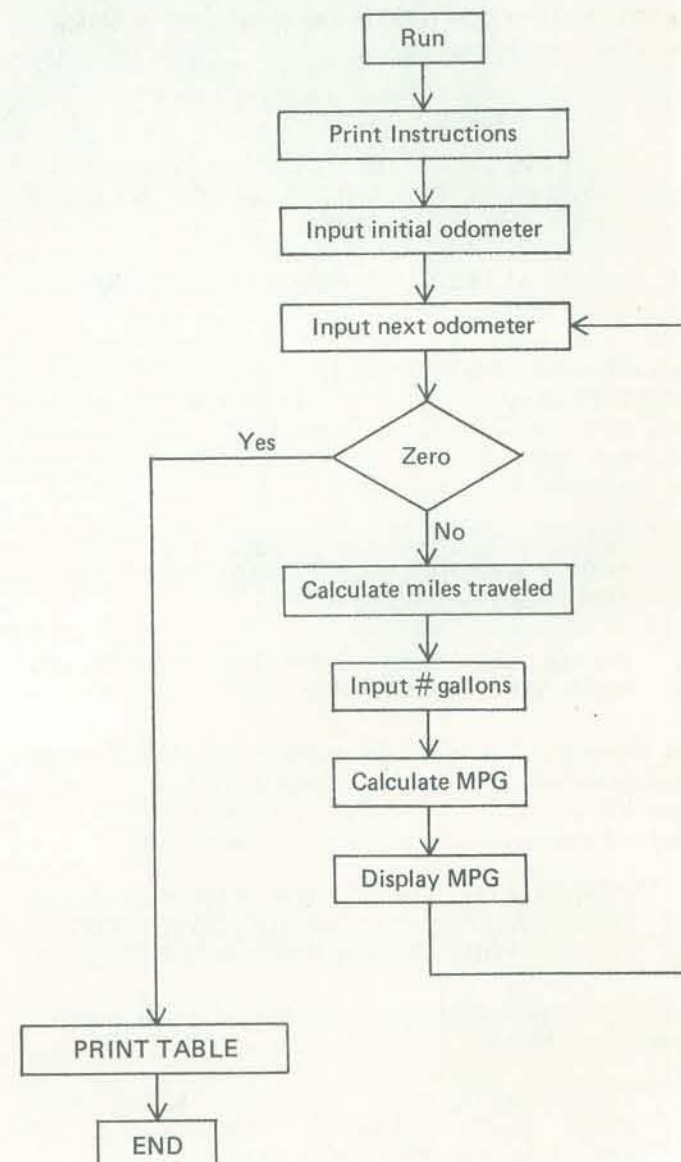
Alternatively, steps two and three can be combined to read $X=A*10+B$

Printing a decimal number is almost as easy. Essentially, our goal is to break-up the integer number with a decimal point inserted at the appropriate location. Assuming the integer 123 is still in X, the following steps will print out the original decimal number:

```
A=X/10      sets A=12
C=A*10      sets C=120 (the 3 has been dropped)
B=X-C       sets B=3
PR A;".";B  prints 12.3
```

You can even eliminate variable C by combining lines two and three to read $B=X-A*10$.

To demonstrate a semi-practical use of this technique, I wrote the following program which will calculate a car's gas mileage. The raw data is collected by filling the tank and noting the odometer reading; then, each time you stop for gas, write down the number of gallons it takes to fill the tank and the current odometer reading. The program takes the sequential



odometer readings and calculates the number of miles traveled between each fill-up. This figure is then divided by the number of gallons consumed, to yield the MPG. The sequence of steps is shown in the flow chart.

The practicality of this program is limited by two things: first,

the highest odometer reading that can be accepted is 3276.7, and second, the program does not round off to the nearest tenth when it calculates the MPG (for example, 20/3 or 6.666... would be listed as 6.6, not as 6.7). In spite of these limitations, the program does illustrate a simple method of manipulating decimal numbers with an integer Tiny Basic. □

CALCULATE YOUR MPG

```

10 CLRS
11 DIM O(50), M(50), G(50), R(50)
12 PR" *** MPG PROGRAM ***"
13 PR:PR"ALL DATA MUST BE TO THE NEAREST TENTH"
14 PR:PR"INPUT REFERENCE ODOMETER READING";
15 GOSUB 100
16 I=1:O(I)=X
20 I=I+1:PR:PR"NEXT ODOMETER READING";
21 GOSUB 100
22 IF X=0 GOTO 200
23 O(I)=X
24 M(I)=O(I)-O(I-1)
25 PR"# OF GALLONS";
26 GOSUB 100
27 G(I)=X
28 R(I)=10*M(I)/G(I)
29 PR "MPG = ";
30 X=R(I):GOSUB 150
31 PR
32 GOTO 20
100 IN A,B
101 IF A>3276 PR"TOO LARGE!":GOTO 100
102 IF B>9 PR"ONLY TENTHS, PLEASE!":GOTO 100
103 X=A*10+B
104 RET
150 A=X/10:B=X-A*10
151 PR A;".";B,"";
152 RET
200 CLRS:J=1-1:I=1
201 PR"      MILES      #GAL      MPG"
203 I=I+1:IF I>J END
204 X=O(I):GOSUB 100
205 X=M(I):GOSUB 150
206 X=G(I):GOSUB 150
207 X=R(I):GOSUB 150
208 PR
209 GOTO 203
  
```

20 Calculates miles traveled and MPG, which is printed out for each entry

100 Inputs data

150 Prints data

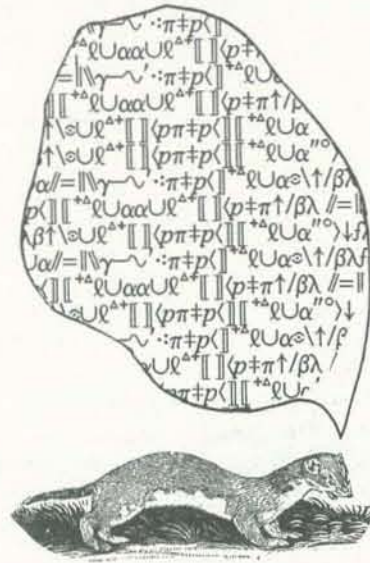
200 Prints table of all data when 0.0 is entered for the next odometer reading

LIST OF VARIABLES

A, B used to input or print decimal data
 X holds integer data for I/O
 I, J index counters
 O() odometer reading
 M() miles traveled
 G() gallons of gas consumed
 R() result as miles per gallon



APL as a Tiny Language



BY MOKURAI CHERLIN

It may seem a little strange to think of APL as a 'tiny' language, but times are changing and with companies such as Microsoft designing and implementing such languages as APL for microcomputers, it won't be long... (yeah, yeah!). We have received a number of articles concerning APL—comparing it with other languages and exploring its potential for various applications—so we will be featuring more APL articles in the future issues.

Reverend Mokurai Cherlin is a Buddhist Priest who lives way up on Mt. Shasta in Northern California and moonlights as a programmer for his father's company, APL Business Consultants, Inc. In our May-June issue of PC, Mokurai painted a general picture of the APL language, describing a little of its history, philosophy, and features. In this article he examines APL with regards to its suitability as a tiny language. —BK

'Tiny Language Talk' in the May-June issue of *People's Computers* contained a proposal by Sam Hills for 24 features that would be desirable in a programming language used by children. Taken together these features provide a power and

flexibility that one might not expect to see even in large languages, and which certainly does not exist in FORTRAN, COBOL or BASIC. This power and flexibility can be found in APL, which satisfies all of Mr. Hills's requirements except those concerning datatypes and declarations. Indeed, one version of APL includes a set of ALGOL-like control structures—Hewlett-Packard's APLGOL is an extension of their APL\3000 which was derived from IBM's APLSV (shared variables). Other APLs have branching and looping controlled by a branch to the value of an expression, a device that provides great flexibility and power, but can be difficult to read.

At present, APLGOL is available only on HP 3000 minicomputers. Another version of APL is being written by Microsoft for 8080 and Z-80 computers, and other dialects have appeared or will appear this year for micros. The Digital Group announced an incomplete APL at the National Computer Conference in June, and has said that the remaining functions will be added in the near future.

I will now describe the features of APL, following the order of Mr. Hills's article.

1. NAMES.

Variable names in APL can be of any length, with the first 77 characters recognized. They must begin with any alphabetic character or Δ , either of which may be underlined, and the rest of the name must be made up of these same characters with the addition of numeric characters. All identifiers (function, variable, group, workspace and file) follow the same pattern, although for some, only 11 characters are recognized. The following are examples of legal APL names:

FNNAME, *LOOP1*, *\Delta*99, *FE*,
NEXTLINE, *FILEFN*

2. LINE NUMBERS.

The system supplies line numbers in user-defined functions. Renumbering is automatic on insertion or deletion of lines. Labels can be used for branch addressing both for clarity and to avoid changing branches throughout the program when modifying it. Using labels can result in a minor increase in execution speed.

3. DATA TYPES.

The user sees data as character or numeric values, with up to 63 dimensions. Most primitive functions operate directly on arrays of any dimension without any need to specify loops for element-by-element processing. Internally numeric data are stored as Boolean (1 bit), integer (2 or 4 byte), or floating point (8 byte). Character data are stored in a 1 byte format. Type conversions are handled automatically for numeric data. One or more dimensions of an array may be of length 0, permitting lists and tables to be initialized empty. APL is ideally suited to graphics displays, since the entire display can be treated as a single matrix, or a set of display matrices can be laminated together in a three dimensional array and referenced using a single coordinate. Color names and color arithmetic present no problems.

4 3.142 7.332E4
AVECTOR OF THREE NUMBERS
'Q?WwEεRp'
AVECTOR OF 8 CHARACTERS

4. CONTROL STRUCTURES.

APLGOL has a full set of control structures; all other versions of APL have only one branch method, but that one is capable of anything that can be done in any other language. In addition, there are other ways to control what is done which require no branching at all. For example, a character string can be constructed using a variety of selection functions to determine the successive substrings, and the string can then be converted into instructions and executed, all in a single line. Character to instruction conversion is carried out either by the execute function or the fix function, the former for single lines and the latter for functions of any size.

APL
→(EXP/LABEL1), LABEL2 *IF*

LOOP:→(∼EXP)/CONTINUE *ADO*
PROCESS
→*LOOP*
CONTINUE

*APL*GOL
IF EXP THEN *PROCEDURE1*
ELSE *PROCEDURE2*

WHILE EXP *DO PROCESS*

5. PROCEDURE CALLS.

APL functions, whether primitive or user-defined, can take 0, 1 or 2 arguments and return 0 or 1 results of any data type, shape or size. Functions which do not return results to be used by other functions usually are intended to cause the printing of some sort of output, or some sort of change to global variables.

No error checking is done at the time of function definition. Programmers need to learn to write provably correct programs, since no amount of error checking or testing can ensure correctness, especially if the program runs but gives wrong results.

Functions can be defined, stored and retrieved in many different ways according to need or convenience. A function can be stored in a workspace and loaded with the workspace or copied into another workspace; it can be converted to character data and stored in a file, then retrieved and converted back into a function. Many operations which require subroutines in other languages are primitive functions in APL.

VR←*CODE*; *KEY*; *TEXT*

[1] *AE*NCRIPTION *PROGRAM*
[2] *AC*CEPTS *KEY* AND *TEXT*
[3] *AF*ROM *KEYBOARD* AND
[4] *AA*SSIGNS THEM TO *LOCAL*
[5] *AV*ARIABLES USING
[6] *AP*ROCEDURES *GET*TEXT
[7] *AA*ND *GET*KEY.
[8] *IN*STRUCTIONS
[9] *TEXT*←*GET*TEXT
[10] *KEY*←*GET*KEY
[11] *R*←*KEY* *PROCESS* *TEXT*
▽

6. LOCAL VARIABLES AND SUBROUTINES.

Variables, including system variables, and user-defined functions can be declared local to a function. A local function is created under program control and disappears from the workspace after the calling function terminates in the same manner as a local variable. This is particularly useful for functions that operate on functions (e.g. differentiation), and is also of value in applications that are too large to fit in a workspace all at the same time.

7. RECURSION.

Recursion is a standard APL feature.

VR←*FACTORIAL* *N*
[1] *ARE*CURSION *EXAMPLE*
[2] *AUSE* !*N* *FOR* *PRODUCTION*
[3] *R*←1
[4] →(*N*=1)*p*0
[5] *AEXIT* *WHEN* *DONE*
[6] *R*←*N*×*FACTORIAL* *N*-1
▽

8. COMMENTS.

Comments are also a standard feature of APL, with different conventions in different dialects. APLSV allows comments on separate lines, APL*PLUS at the end of any line, and APL\3000 anywhere.

[1] *APL* *COMMENT*
[2] *⊕A* *APL***PLUS* *COMMENT*
[3] *APL*GOL *COMMENT*A

9. INITIALIZATIONS & CONSTANTS.

Since there are no declarations in APL and since type, shape and size are all dynamically variable, initialization is the same as assignment. A variable which is never respecified acts as a constant.

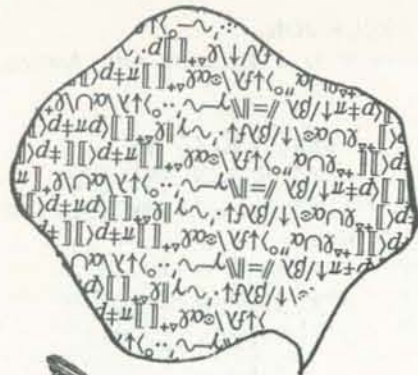
LIST←0 10*p*''
*AE*MPTY *CHARACTER* *MATRIX* *OF*
ATEN *COLUMNS*
LIST←*LIST*, 'ABCDEFGHIJ'
APPEND *NEW* *ROW*

10. STANDARD TYPES.

Kenneth Iverson had data types in mind when he created the notation which developed into APL. The primitive types are just arrays of any dimension, but all other data structures in common use are easily defined in terms of them. Iverson's original book, *A Programming Language*, shows in detail how to do this very simply. Some work is being done on non-uniform arrays with a view to incorporating them into APL (RECORD type with mixed character and numeric data for example).

11. SUBRANGES OF ARRAYS.

Subarrays of any dimension can be assigned.



```
AR←3 4p112
AR
1 2 3 4
5 6 7 8
9 10 11 12
AR[1 2;2]←-3
AR
1 -3 3 4
5 -3 7 8
9 10 11 12
```

12. CONCATENATION OF STRINGS.
Concatenation is a primitive APL function. It can be applied to arrays of any size whose dimensions are suitable, to join them either along an existing coordinate or a new one, and either character or numeric arrays may be concatenated.

```
A←'HI'
B←'THERE!'
A,' ',B
HI THERE!
```

13. INPUT ERROR RECOVERY.
APL has two input functions, which trap character errors themselves and permit the programmer to trap any error he can program for, with any response he may desire.

Quad input treats a line of text up to a carriage return as an expression to be evaluated. It rejects a blank line and gives another prompt. The function containing the quad is given the value of the expression and can check it for range etc. before acting on it.

Quote-quad input takes an input line as a string of characters, permitting it to be checked before anything at all is done with it. Thus, a numeric valued expression can be checked for syntax errors, then converted to numeric form and checked again for range, shape, etc. Use of quote-quad input prevents any use of the system except that written into the program. In quad input, the evaluation of the expression is uncontrolled, and thus a clever user may get inside the program and meddle.

```
VANSWER←VERIFYΔINPUT
[1] APROMPT
[2] 'TYPE ANSWER'
[3] AACCEPT INPUT;CHECK
[4] READ:→(~OK INPUT+▽/ERR
OR
[5] ANSWER←ΔINPUT
[6] AINPUT IS CONVERTED
[7] →0
[8] ERROR:'PLEASE TYPE MORE CAREFULLY'
[9] →READ
▽
```

14. AUTOMATIC STRING/NUMERIC CONVERSION.
A primitive APL function, execute, permits any character string representing an expression to be evaluated. Type conversion from character to numeric depends on this function and is not automatic.

15. DECLARATION OF VARIABLES.
Variables are not declared in APL. Spelling errors are usually caught when a variable having no value is referenced, resulting in a VALUE ERROR during execution.

16. PRETTY OUTPUT.
APLgol formats functions according to structure. If ordinary APL functions are written using only two branch structures (corresponding to IF and DO WHILE), a formatting function can easily be written to recognize them and format the character representation of the function.

17. EDITING.
All APLs have function editing capabilities, including line insertion/deletion and replacement, and character insertion/dele-

tion and replacement. Some APLs have extended editors with other capabilities such as retrieval and editing of immediate mode input. Text editors of any degree of sophistication can be readily implemented in APL and are available commercially from a number of sources. Blank lines are not admissible in functions, but they are not needed for separation of modules, which can be written as separate functions called by a master program.

```
[5] VAR1+VAF3
,VAR2+
[5] VAR1+VAR2+VAF3
/,R
[5] VAR1+VAR2+VAR3
[6] [~6]
[7] ▽
AINSERTION, REPLACEMENT,
ADELETION OF LINE, LEAVE
AFUNCTION EDIT MODE
```

18. STRING-TO-NUMERIC CONVERSION.

See point 14 above. APL also provides numeric-to-string conversion through the format primitive function. This function can also be used to format matrix data for printing, with full control of number of digits and placement. Some APLs have more elaborate formatting functions suitable for preparing business reports and the like.

```
A←3
B←'ANSWER IS: '
B;A
ANSWER IS: 3
AMIXED OUTPUT, CHARACTER
AND NUMERIC ON SAME LINE
R←B,VA
R
ANSWER IS: 3
AA CONVERTED TO CHARACTER
AND CONCATENATED TO B;
ARESULT CAN BE ASSIGNED
```

19. STANDARD FUNCTIONS AND PROCEDURES.

APL contains arithmetic, log, trig, hyperbolic, exponential, comparison, logical, selection (from arrays), sort, random number, matrix multiply, divide and in-



verse, change base, factorial and Gamma, binomial coefficient and Beta, and many more primitive functions. Function-to-character, character-to-function, workspace and file information and control system functions, notably the facility to load a new workspace under program control. A statement called the 'latent expression' can be set to begin execution automatically on loading, thus providing for the chaining together of workspaces, which can communicate through the file system.

```
ARANDOM NUMBER
?5 5 5
3 1 4
AFACTORIAL
!1 2 3 4
1 2 6 24
AREXPONENTIAL
2 3 4*4 3 2
16 27 16
2*8 9 10
256 512 1024
AMATRIX INVERSE
A←2 2p1 1 0 1
A
1 1
0 1
A
1 -1
0 1
```

21. STRING MATCHES AND SUBSTITUTION.

These operations are easily implemented using vector operations. On at least one system, string matching is available as a system function.

20. DYNAMIC TYPES.
Dynamic types are standard. On some systems file sizes have to be declared (usually tape-based systems); on others, file size and type are completely dynamic.

```
'WHY DON'T YOU?'SS'DO'
0 0 0 0 1 0 0 0 0 0 0 0 0 0
ASTRING SEARCH SYSTEM
AFUNCTION; APL*PLUS ONLY
```

22. CALCULATOR MODE.
This is also standard and does not require any indicator; rather a special symbol is used to enter and leave function definition mode.

23. JOYSTICKS.
Such accessories are readily implemented through shared variables or other means.

24. MACHINE-LANGUAGE SUB-ROUTINES.
Machine-language subroutines are available in APL 3000 and APLGOL, but not necessarily in other APLs.

CONCLUSION.
All the features desired by Mr. Hills are available in one version or another of APL, (along with lots of other goodies) except for the few dealing with data declarations. We are not likely to have an APL available on micros soon with anywhere near all the extras described here built in; but the only feature that Mr. Hill asked for which could not be easily provided in a defined function is the ability to call machine language subroutines.

Microsoft's promised APL will apparently run on a TRS-80, Level II using the T-Bug monitor to enter and call the APL interpreter. To do so would be cumbersome, but not as bad as what BASIC makes you do while programming. If the market could be demonstrated, so that mask-programmed ROMs could be used, existing technology as seen in the PET computer could be used to create a self-contained APL system priced between \$1000 and \$1500. Almost the entire difference in price may be attributed to the necessity of about 20k of extra ROM for the interpreter. This is not a small language, and for the next few years it will not be cheap. A minimum TRS-80 system would cost about \$2100. A minimum 8080 system with

an APL terminal and floppy disk will go for about \$3000 on the day the interpreter is released.

Next year we will have APL with virtual memory and timesharing on Z-8000 and 8086 16 bit microcomputers. In a school situation, this will bring the price per station down to about \$1500 (terminal plus memory plus a share of the base system). This is possible because 8080 code can be translated automatically into both 8086 and Z-8000 code. This is inefficient, since it will not make use of any 16 bit instructions; however it provides breathing space to rewrite the interpreter while the users get on with their business. Grosch's law states that twice the money can buy four times the computing power; this has been verified often in large systems, and it also seems to be true in the small computer realm. This can give schools considerable leverage in providing computing power for students. Indeed, \$1500 per student for a class of 30 would buy several megabytes of disk storage per student and workspaces on the order of 10K bytes per student at full load using one sufficiently fast processor.

The APL language is being used in the public schools of several cities as the ordinary language for instruction in mathematics, without reference to its use as a computer language. In a few places it is used on computers for lecture and demonstration purposes, and even for student programming. Its virtues as an instructional language are greater clarity than standard notation, executability and suitability for graphics. A display can be treated as a matrix, to be operated on with all the array handling power of the language. Equations can be graphed as fast as they are entered at the keyboard. The effects of parameters can be demonstrated in the motion of a curve calculated at high speed. Such things can be done in any language, but APL makes it trivially easy, so that less expensive software (apart from the interpreter) is needed, and changes can be made at will. For learning mathematics and mathematical subjects by experiment, there is no language which can approach the simplicity and power of APL (even though PASCAL is probably a better language for teaching the discipline of programming), and none that go as far to let the student learn rather than struggle. □

• Over 250 Exhibit Spaces

• Internationally Recognized Speakers

• Held in the VAST Dallas Convention Center

• Special Programs for Dealers Only

Dallas-Sept. 29-30-Oct. 1, 1978

**EXHIBITORS
(AS OF JUNE 10)**

ADVANCED COMPUTER PRODUCTS, ALPHA MICRO SYSTEMS, APPLE, APPLIED DATA COMMUNICATIONS, AXIOM CORP., BYTE SHOP OF DALLAS, CAPITAL EQUIPMENT BROKERS, CENTRONICS, COMPUCOLOR CORP., COMPUTER HEADWARE, COMPUTER ROOMERS, COMPUTER SHOP, DALLAS COMPUTER CENTER, DATA GENERAL CORP., DECISION DATA COMPUTER CORP., DIGITAL EQUIPMENT CORP., DIGITAL RESEARCH CORP., DILITHIUM PRESS, DIVERSIFIED TECHNOLOGY, D P SERVICES, ELECTRONIC DATA SYSTEMS (EDS), FINANCIAL COMPUTER CORP., FOUNDATION FOR QUALITY EDUCATION, GENERAL ELECTRIC, GIMIX, INC., GODBOUT ELECTRONICS, HOBBY WORLD ELECTRONICS, IMSAI, ITHACA AUDIO, JADE COMPUTER PRODUCTS, K A ELECTRONICS, METROPLEX DATA SYSTEMS, INC., MICRO AGE, MICRO DIVERSION, INC., MICROPOLIS CORP., MIDWEST SCIENTIFIC INST., MITS, MOSTEK, MOTOROLA SEMICONDUCTOR, NDAKES DATA COMMUNICATIONS, NORTH TEXAS COMPUTER CLUB, O K MACHINE & TOOL, OSBORNE & ASSOCIATES, PAGE DIGITAL ELECTRONICS, PERTEC MICRO SYSTEMS, PERCOM DATA CORP., PRIME SUPPLY, INC., PROBLEM SOLVER SYSTEMS, Q M DATA SERVICE, QUALITY COMPONENTS, QUEST ELECTRONICS, RADIO HUT, S D SYSTEMS, SCHWEBER ELECTRONICS CORP., SEALS ELECTRONICS, INC. SMOKE SIGNAL BROADCASTING, SOUTHWEST FEDERATION OF COMPUTER CLUBS, SOUTHWEST TECHNICAL PRODUCTS, SPACE BYTE, SUMMAGRAPHICS CORP., SYBEX, INC., SYNERTEK, TANDY CORP., TECHNOCORP., TEKTRONIX INDUSTRIES, TELPAR, INC., TEXAS INSTRUMENTS, 3M COMPANY, V R I, VANGUARD SYSTEMS CORP., VECTOR GRAPHICS, INC., WEST & ASSOCIATES, XEROX CORP., ZITEX CORP.

**SEE TOMORROW
—TODAY!**

**International
Microcomputer
Exposition**

Keynote Address By Dr. Portia Isaacson

Special Dealer Program

Featured Seminar Speakers

| | | | |
|-----------|-------|-----|--|
| Name | Title | | |
| Company | | | |
| Address | | | |
| City | State | Zip | |
| Telephone | | | |

Advance Registration

One Day Admission \$4 (at door \$6) _____
 Three Day Admission \$8 (at door \$10) _____
 Seminar Admission \$15 _____
Total _____

Make Checks payable to I.M.E. — 413 Carillon Tower — 13601 Preston Road — Dallas, Texas 75240 214/271-9311

SPEAKERS

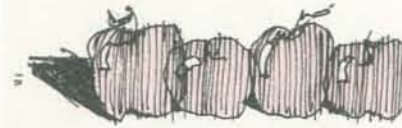
CAROL OGDIN (SOFTWARE TECHNIQUES), DR. ADAM OSBORNE (OSBORNE & ASSOCIATES), WAYNE GREEN, (KILOBAUD MAGAZINE), CHRIS MORGAN (BYTE MAGAZINE) BOB JONES (INTERFACE AGE), ZACH BOVINETTE (INTERFACE AGE), STEVE MURTHA (D/A ASSOCIATES), ELLIOT MAC LENNAN (MAC LENNAN & LILLIE), ASTRONAUT (NASA), HAROLD MAUCH (PERCOM DATA), ELIZABETH JACKSON (SOFTWARE TECHNIQUES), BOB ELDRIDGE (DIGITAL EQUIPMENT CORP.), RODNEY ZAKS (SYBEX), DR. EMERSON BROOKS, (E SYSTEMS), R. NEIL FERGUSON (MOORE BUSINESS FORMS), GEORGE NELSON (MOTOROLA), BEN PEEK (BEN PEEK, INC.), STEVE EDELMAN (ITHACA AUDIO), BILL GODBOUT (GODBOUT ELECTRONICS), JOHN E. HOWLAND (VANGUARD SYSTEMS CORP.), MITCH GOOZE (MOTOROLA SEMICONDUCTOR PRODUCTS, INC.), BOB FULLER (TEXAS INSTRUMENTS), PHILLIPE de MARCHIN (FAIRCHILD), DAVID AHL (CREATIVE COMPUTING), S. PAL ASIJA (ASIJA LAW OFFICE), DR. THOMAS J. BLACK (SOLAR-STATE SYSTEMS), DANIEL D. HAMMOND (SD SYSTEMS), JOHN P. SMITH (SCHWEBER ELECTRONICS), GEORGE MORROW (THINKER TOYS), HOWARD FULMER (PARASITIC ENGINEERING), DR. RICHARD HODGES (U.T.D.), NORMAN REITZEL (FORMERLY DATAPoint), DR. AARON H. KONSTAM (TRINITY UNIVERSITY), D.C. DEFFENBAUGH (HOME COMPUTER CENTER, INC.), HOWARD J. HILTON.

**MAGAZINES
EXHIBITING**

COMPUTER DEALER, COMPUTER RETAILING, CREATIVE COMPUTING, INTERFACE AGE, KILOBAUD, POPULAR ELECTRONICS, RADIO ELECTRONICS, SMALL BUSINESS COMPUTER.

APPLE MATH

BY JOHN GAINES



John Gaines noticed that up till now, we haven't published any programs for the Apple II. He assumed (correctly) that this was due to the fact that none were being submitted, and he decided to start the apple rolling by sending us this math drill. We regret that we cannot bring you this program in living color, but at least we have a juicy red apple for an illustration. If you are interested in swapping Apple II programs with John Gaines or if you want a cassette tape of Apple-Math (\$10) write to John at at 7407 Meadow Hill, San Antonio, TX 78251. — BK

I initially got the idea to write this program after hearing my wife complain how hard it was to get her 3rd graders interested in learning math. It started out as a very simplistic program and was not very interesting. Then the brainstorm hit me; I stayed up until almost 5 am, adding the menu, the introduction, and the color reinforcement, and then I spent the next 3 to 4 weeks debugging it and simplifying it so that small children could read and understand it.

There are many uses in the home for the Apple II and I feel that the most neglected use is that of educating your children. It offers them a different and challenging way to learn. You will be amazed at how the Apple will get your children interested in studying again when all other ways have failed.



HI, I'M THE APPLE II COMPUTER AND I AM GOING TO HELP YOU LEARN YOUR MATH. WHEN YOU SEE THE FLASHING SQUARE, I AM ASKING FOR YOUR ANSWER

JUST TYPE IT IN AND HIT THE 'RETURN' KEY ON THE RIGHT. TO START PRACTICING JUST HIT RETURN.

HELLO AGAIN, WHAT IS YOUR NAME ?JOHN

THE APPLE II SCHOOLHOUSE MATH MENU WHAT KIND OF PROBLEM DO YOU WANT, JOHN

1. ADDITION
2. MULTIPLICATION
3. SUBTRACTION
4. DIVISION

ENTER THE NUMBER BESIDE THE TYPE OF PROBLEM YOU WANT TO DO AND

HIT THE RETURN KEY, NUMBER ?1

```

228
+ 24
----
???
```

YOUR ANSWER ?252

```

255
+ 18
----
???
```

YOUR ANSWER ?273

HOW ABOUT SOME MORE PROBLEMS? (YES-NO)NO

HEY, JOHN

DON'T GIVE UP NOW...

KEEP PRACTICING AND YOU'LL GET BETTER!

AFTER ALL YOU'RE ONLY HUMAN.....

I'M AN APPLE II COMPUTER!

IF YOU WANT ANOTHER PROBLEM, TYPE 'OK'.OK

THE APPLE II SCHOOLHOUSE MATH MENU WHAT KIND OF PROBLEM DO YOU WANT, JOHN

1. ADDITION
2. MULTIPLICATION
3. SUBTRACTION
4. DIVISION

ENTER THE NUMBER BESIDE THE TYPE OF PROBLEM YOU WANT TO DO AND

HIT THE RETURN KEY, NUMBER ?2

```

  15
x 34
----
???
```

YOUR ANSWER ?510

```

  43
x 61
----
???
```

YOUR ANSWER ?2623



INSTRUCTIONS FOR USING APPLE MATH

To change the level of difficulty just follow the examples listed below with the appropriate line numbers.

ADDITION

Change statements 2020 to 2035
2020 Let A=RND (999)

This determines the value of 'A', in this case it is a 3 digit number not over 999. You can change the (999) to (99) for example and it will be a two digit number.

2025 If A < 99 then 2020

This will not allow 'A' to be less than a three digit number. You must change to < 9 if you make 'A' (99) and change to < 1 if you make 'A' (9)

Use the same format to change the variables for 'B'.

MULTIPLICATION

Change statements 3210 to 3220 and use the same instructions as above. CAUTION: You cannot have any answer larger than 32767

SUBTRACTION

Change statements 4005 to 4020 and use as above.

DIVISION

Change statements 8520 to 8530 and use as above.

NOTE: The print statements that follow each variable may need a little format adjustments to line the numbers up correctly after you change the variables.

To change the number of problems displayed in a row:

ADDITION

Statement 2015 For Z=1 to 2 (will go thru two problems) change to 2015 For Z=1 to 10 (will go thru 10 problems)

MULTIPLICATION

change statement 3205 For G=1 to 2

SUBTRACTION

Change statement 4000 For F=1 to 2

DIVISION

Change statement 8515 For P=1 to 2



Apple Math continued...



```

100 CALL -936
110 DIM A$(20)
120 DIM C$(4)
130 DIM D$(4)
140 DIM E$(4)
150 DIM F$(4)
200 GOTO 5000
1005 PRINT : PRINT : PRINT
1010 PRINT "HEY ",A$
1015 PRINT : PRINT "DON'T GIVE UP NOW..."
1020 PRINT : PRINT "KEEP PRACTICING AND YOU'LL GET BETTER!"
1025 PRINT : PRINT "AFTER ALL YOU'RE ONLY HUMAN...": PRINT
1030 PRINT : PRINT "I'M AN APPLE II COMPUTER!": PRINT : PRINT
      : PRINT : PRINT : PRINT
1035 INPUT "IF YOU WANT ANOTHER PROBLEM, TYPE 'OK'.",D$
1040 IF D$="OK" THEN 6000
1045 GOTO 5000
1050 CALL -936: VTAB 12
1055 PRINT "WELL, ";A$;": PRINT : PRINT "I GUESS I'M JUST TOO MUCH FOR YOU."
      : PRINT
1060 PRINT "COME SEE ME AGAIN...WHEN YOU THINK YOUR": PRINT "UP TO IT.":
      PRINT : PRINT " " " " "APPLE II"
1065 PRINT : PRINT " " " " "APPLE II"
1070 FOR X=1 TO 3000: NEXT X
1075 GOTO 6000
1815 FOR B=0 TO 1
1900 TEXT : CALL -936: GR
1905 COLOR=3: HLIN 1,39 AT 1: HLIN 1,39 AT 39: VLIN 1,39 AT 1: VLIN 1,39
      AT 39
1910 COLOR=14: HLIN 10,32 AT 17: HLIN 10,32 AT 18: HLIN 10,32 AT 19: HLIN
      10,32 AT 20
1950 HLIN 17,25 AT 12: HLIN 17,25 AT 13: HLIN 17,25 AT 14: HLIN 17,25 AT
      15: HLIN 17,25 AT 16: HLIN 17,25 AT 25
1960 HLIN 17,25 AT 26: HLIN 17,25 AT 27: HLIN 17,25 AT 28: HLIN 17,25 AT
      29
1970 FOR T=1 TO 1000: NEXT T
1980 GOTO 2010
2000 GOTO 6000
2010 TEXT : CALL -936
2015 VTAB 12: FOR Z=1 TO 2
2020 LET A= RND (999)
2025 IF A<99 THEN 2020
2030 LET B= RND (99)
2035 IF B<9 THEN 2030
2040 LET C=A+B
2045 PRINT "
      + "
2050 PRINT "
      + "
2055 PRINT "
      ???"
2060 PRINT "
      ???"
2065 PRINT : INPUT "YOUR ANSWER ",C1
2070 IF C1=C THEN 2099
2075 FOR N=1 TO 100
2080 N1= PEEK (-16336)+ PEEK (-16336)+ PEEK (-16336)
2085 NEXT N
2090 GOSUB 7000
2095 PRINT "
      ";A;"+";B;"+"?"
2097 GOTO 2065

```

```

2098 TEXT : CALL -936
2099 GOSUB 8000
3000 VTAB 12
3005 NEXT Z
3010 VTAB 12: CALL -936
3020 INPUT "HOW ABOUT SOME MORE PROBLEMS? (YES-NO)",C$
3025 IF C$="YES" THEN 6000
3030 GOTO 1000
3035 GOTO 6000
3050 TEXT : CALL -936: GR
3053 COLOR=7: HLIN 1,39 AT 39: HLIN 1,39 AT 1: VLIN 1,39 AT 39: VLIN 1,39
      AT 1
3055 COLOR=4: HLIN 10,16 AT 13: HLIN 11,17 AT 14: HLIN 12,18 AT 15: HLIN
      13,19 AT 16: HLIN 14,20 AT 17: HLIN 15,21 AT 18: HLIN 16,22 AT 19: HLIN
      17,23 AT 20
3060 HLIN 18,24 AT 21: HLIN 19,25 AT 22: HLIN 20,26 AT 23: HLIN 21,27 AT
      24: HLIN 22,28 AT 25: HLIN 23,29 AT 26: HLIN 24,30 AT 27
3065 HLIN 25,31 AT 28: HLIN 26,32 AT 29: HLIN 26,31 AT 13: HLIN 25,30 AT
      14: HLIN 24,29 AT 15: HLIN 23,28 AT 16: HLIN 22,27 AT 17
3070 HLIN 21,26 AT 18: HLIN 20,25 AT 19: HLIN 19,24 AT 20: HLIN 18,23 AT
      21: HLIN 17,22 AT 22: HLIN 16,21 AT 23: HLIN 15,20 AT 24
3075 HLIN 14,19 AT 25: HLIN 13,18 AT 26: HLIN 12,17 AT 27: HLIN 11,16 AT
      28: HLIN 10,15 AT 29
3080 FOR T=1 TO 1000: NEXT T
3085 GOTO 3200
3200 TEXT : CALL -936
3205 FOR G=1 TO 2
3208 VTAB 12
3210 LET M= RND (99)
3213 IF M<9 THEN 3210
3220 IF N<=9 THEN 3215
3225 PRINT "
      ??"
3230 PRINT "
      ??"
3235 PRINT "
      ??"
3240 PRINT "
      ??"
3245 PRINT : INPUT "YOUR ANSWER ",Z1
3250 LET Z=M*N
3253 IF Z1=Z THEN 3290
3255 FOR B=1 TO 100
3260 B1= PEEK (-16336)+ PEEK (-16336)+ PEEK (-16336)
3265 NEXT B
3270 GOSUB 7000
3275 PRINT "
      ";M;"X";N;"?"
3280 GOTO 3245
3285 TEXT : CALL -936
3290 GOSUB 8000
3295 NEXT G
3300 GOTO 3010
3305 TEXT : CALL -936
3800 TEXT : CALL -936: GR
3810 COLOR=4: HLIN 1,39 AT 39: HLIN 1,39 AT 1: VLIN 1,39 AT 39: VLIN 1,39
      AT 1
3820 COLOR=9: HLIN 10,31 AT 16: HLIN 10,31 AT 17: HLIN 10,31 AT 18: HLIN
      10,31 AT 19: HLIN 10,31 AT 20: HLIN 10,31 AT 21: HLIN 10,31 AT 22
3830 FOR H=1 TO 1000: NEXT H
3840 GOTO 4000
4000 TEXT : CALL -936: FOR F=1 TO 2
4003 CALL -936: VTAB 12
4005 LET O= RND (999)
4010 IF O<99 THEN 4005
4015 LET T= RND (99)
4020 IF T<9 THEN 4015
4025 LET U=O-T

```

```

4030 PRINT "
      -"
4035 PRINT "
      -"
4040 PRINT "
      -"
4045 PRINT "
      ???"
4050 PRINT : INPUT "YOUR ANSWER ",U1
4055 TEXT : CALL -936
4060 IF U1=U THEN 4095
4065 FOR N=1 TO 100
4070 N1= PEEK (-16336)+ PEEK (-16336)+ PEEK (-16336)
4075 NEXT N
4080 GOSUB 7000
4085 PRINT "
      ";O;"-" ;T;"="?"
4090 GOTO 4050
4095 GOSUB 8000
4100 NEXT F
4105 GOTO 3010
5000 CALL -936
5005 VTAB 6: PRINT "HI, I'M THE APPLE II COMPUTER AND I AM": VTAB 8: PRINT
      "GOING TO HELP YOU LEARN YOUR MATH."
5010 VTAB 10: PRINT "WHEN YOU SEE THE FLASHING SQUARE, I AM": VTAB 12: PRINT
      "ASKING FOR YOUR ANSWER": PRINT
      : CALL -380: PRINT "KEY ON THE RIGHT."
5020 PRINT "JUST TYPE IT IN AND HIT THE "; CALL -384: PRINT "RETURN"
      : CALL -380: INPUT "TO START PRACTICING JUST HIT RETURN.",V$
5030 VTAB 20: INPUT "
      HELLO AGAIN, ": VTAB 14: INPUT "WHAT IS YOUR NAME?",A$
5060 GOTO 6000
6000 CALL -936: VTAB 5
6010 CALL -384: PRINT " THE APPLE II SCHOOLHOUSE MATH MENU ": CALL -380
      : PRINT
6020 VTAB 8: PRINT "WHAT KIND OF PROBLEM DO YOU WANT.",A$
      : PRINT
6040 TAB 7: PRINT "3. SUBTRACTION": PRINT : TAB 7: PRINT "4. DIVISION"
6050 PRINT : PRINT "ENTER THE NUMBER BESIDE THE TYPE OF": PRINT : PRINT
      "PROBLEM YOU WANT TO DO AND": PRINT
6055 INPUT "HIT THE RETURN KEY. NUMBER ",A1
6060 CALL -936
6065 IF A1=1 THEN 1900
6070 IF A1=2 THEN 3050
6075 IF A1=3 THEN 3800
6080 IF A1=4 THEN 8500
7000 GR
7005 COLOR=2
7010 VLIN 5,10 AT 4: VLIN 5,10 AT 5: VLIN 5,10 AT 8: VLIN 5,10 AT 9: VLIN
      5,10 AT 12: VLIN 5,10 AT 13
7015 VLIN 5,10 AT 16: VLIN 5,10 AT 17: VLIN 5,10 AT 20: VLIN 5,10 AT 21:
      VLIN 5,8 AT 23: VLIN 5,8 AT 24
7020 VLIN 5,7 AT 27: VLIN 5,7 AT 28: VLIN 7,10 AT 32: VLIN 7,10 AT 33: VLIN
      5,9 AT 36: VLIN 5,9 AT 37
7025 HLIN 36,37 AT 11: HLIN 4,9 AT 5: HLIN 4,9 AT 10: HLIN 12,17 AT 5: HLIN
      12,17 AT 10: HLIN 20,24 AT 5: HLIN 20,24 AT 8
7030 HLIN 27,32 AT 5: HLIN 27,32 AT 7: HLIN 27,32 AT 10: COLOR=8: HLIN 4
      ,33 AT 12
7035 COLOR=14: HLIN 4,9 AT 16: HLIN 12,16 AT 16: HLIN 14,14 AT 19: VLIN
      17,21 AT 6: VLIN 17,21 AT 7: VLIN 17,21 AT 12
7040 VLIN 17,21 AT 13: VLIN 17,21 AT 15: VLIN 17,19 AT 16: VLIN 16,19 AT
      19: VLIN 16,19 AT 20: VLIN 19,21 AT 21: VLIN 19,21 AT 22: VLIN 16,19
      AT 23: VLIN 16,19 AT 24
7045 COLOR=4: VLIN 25,30 AT 8: VLIN 25,30 AT 9: VLIN 25,30 AT 11: VLIN 25
      ,30 AT 12: VLIN 25,30 AT 15: VLIN 25,30 AT 16
7050 VLIN 25,26 AT 10: VLIN 28,28 AT 10: VLIN 28,30 AT 19: VLIN 25,30 AT
      22: VLIN 25,30 AT 23: VLIN 25,30 AT 25: VLIN 25,30 AT 26

```

```

7055 VLIN 25,26 AT 24: VLIN 28,28 AT 24: VLIN 25,30 AT 29: VLIN 25,30 AT 29: VLIN 25,30 AT
      30: VLIN 25,30 AT 33: VLIN 25,30 AT 34
7060 VLIN 25,30 AT 38: VLIN 25,30 AT 39: VLIN 26,26 AT 35: VLIN 27,27 AT
      36: VLIN 28,28 AT 37
7065 HLIN 10,10 AT 28: HLIN 17,18 AT 30: HLIN 18,19 AT 26: HLIN 18,18 AT
      28: HLIN 15,19 AT 25
7070 COLOR=9: HLIN 8,39 AT 32: COLOR=6: HLIN 8,39 AT 34: COLOR=7: HLIN 8
      ,39 AT 36
7075 RETURN
8000 FOR S=0 TO 1
8005 GR : COLOR=2: VLIN 2,38 AT 3: VLIN 2,38 AT 4: VLIN 2,38 AT 5: VLIN
      2,38 AT 11: VLIN 2,38 AT 12: VLIN 2,38 AT 13
8010 VLIN 2,38 AT 16: VLIN 2,38 AT 17: VLIN 2,38 AT 18: VLIN 2,38 AT 23:
      VLIN 2,38 AT 24: VLIN 2,38 AT 25: VLIN 2,38 AT 28: VLIN 2,38 AT 29
      : VLIN 2,38 AT 30
8015 HLIN 11,18 AT 2: HLIN 11,18 AT 3: HLIN 11,18 AT 4: HLIN 11,18 AT 36
      : HLIN 11,18 AT 37: HLIN 11,18 AT 38
8020 HLIN 23,30 AT 2: HLIN 23,30 AT 3: HLIN 23,30 AT 4: HLIN 23,30 AT 36
      : HLIN 23,30 AT 37: HLIN 23,30 AT 38
8025 COLOR=4: HLIN 1,39 AT 39
8030 FOR XC=1 TO 300: NEXT XC
8035 TEXT : CALL -936
8040 NEXT S
8045 RETURN
8500 GR : COLOR=6: HLIN 10,29 AT 19: HLIN 10,29 AT 18: HLIN 10,29 AT 20:
      HLIN 10,29 AT 21: HLIN 18,21 AT 14: HLIN 18,21 AT 13
8503 HLIN 18,21 AT 25: HLIN 18,21 AT 26
8505 COLOR=13: HLIN 1,39 AT 39: HLIN 1,39 AT 1: VLIN 1,39 AT 39: VLIN 1,
      39 AT 1
8507 FOR T=1 TO 1500: NEXT T
8510 TEXT : CALL -936
8515 FOR P=1 TO 2
8520 X= RND (82)
8525 IF X=0 THEN 8520
8530 Y= RND (10)
8535 IF Y=0 THEN 8530
8540 IF Y=X THEN 8520
8545 IF Y=X*10 THEN 8530
8550 IF X MOD Y<>0 THEN 8520
8560 LET C=X/Y
8565 CALL -936: VTAB 12: PRINT "WHAT IS THE QUOTIENT OF ";X;" / ";Y;": PRINT
      : PRINT "YOUR ANSWER ",C1
8575 IF C1=C THEN 8620
8580 FOR N=1 TO 100
8585 N1= PEEK (-16336)+ PEEK (-16336)+ PEEK (-16336)
8590 NEXT N
8592 GOSUB 7000
8595 PRINT "THE QUOTIENT OF";X;" / ";Y;" = ?"
8600 GOTO 8570
8615 CALL -936: VTAB 20
8620 GOSUB 8000
8625 NEXT P
8630 GOTO 3010
9000 REM "APPLE II COLOR TEACHER" BY JOHN GAINES-COMPUTER SOLUTIONS N.W. SAN
      ANTONIO,TX (512)828-1455

```



SPOT

The Society of PET Owners and Trainers

EDITED BY PHYLLIS COLE



PET photo courtesy of Utter Chaos

Commodore's PET is a factory assembled personal computer based on a 6502 microprocessor. The \$795 system includes a keyboard, cassette tape unit, built-in TV screen, some graphics, upper and lower case, and extended 8K BASIC, and 8K of user memory. Each bimonthly issue of People's Computers since the September-October 1977 issue has included an article on the PET. —PC

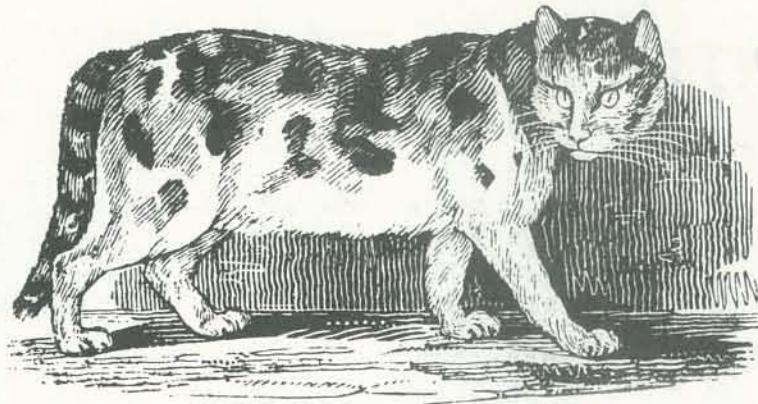
COMMODORE CORNER

At last, the long-awaited guidelines for submission of software for publication by Commodore are available. When/if they reach the hands of those of you who've submitted software is another question. Personally, we've had, at best, erratic responses—usually none—in requesting information. One example: last November, *People's Computers* wrote to Commodore offering to compile a booklet of articles, programs, etc., that could be sent to new owners to provide some pointers about PET usage, since almost no documentation was available. We were prepared to do this at no expense to Commodore, as a way of getting some free advertising and also offering what we saw as a valuable

service. We received no response to the letter.

As indicated in the PET PEEVES section, others have been similarly frustrated in dealing with the company. One reason for this may be the very rapid turnover of personnel. We know of several employees who lasted only a few months; among them is Adrian Byram, the Software Manager whose name appears on the guides to software publication that we've seen.

We're discouraged with the company, though we still enjoy our PET. Perhaps the 'COMMODORE CORNER' can be summed up in a single word: confusing.



PET PEEVES

CONLEY'S COMMENTS

On December 16, 1977, the Reverend David M Conley sent a letter to Commodore detailing a variety of problems that the Universal Life Church of the Pacific was experiencing with its PET, and attempts to set up an appointment to have it repaired. They were understandably reluctant to ship their system back to the factory for an unknown period of time, and they were hopeful of finding some other way of getting their PET to function more satisfactorily. Copies of the letter were sent to 5 publications and an attorney.

Six months later, a second letter was sent by the Reverend Conley to Commodore, pleading for some response. We sincerely hope that by now Commodore has dealt with the problems. One of the most discouraging items in the letters was the fact that Commodore had not, in a 6 month period, even sent out its instruction booklet.

DOCUMENTATION DILEMMA

As indicated above, documentation—rather, the lack thereof—is a severe handicap for PET users. Many purchasers open their new PETs to find not a scrap of information, despite the fact that since last November various booklets, however slim, have in fact been available from the company. One reason given for the lack of documentation was that all copies of the previous booklet had been shipped; and, rather than print more of the old version, the new version would be shipped to these customers at a future date. This sort of explanation is hardly acceptable. As more and more publications try to fill the gap Commodore has left, some users will find that with time, effort and money the desired information can be accumulated. But the process is often slow and painful, and for those unsuspecting persons who buy a PET thinking they're getting an appliance, the process may be next to impossible to initiate. After all, if you've never used a computer, never read a computer magazine, and there's no computer store in your town, just where do you begin?

PET POINTERS

PRETTY PRINTING

The PET left-adjusts print functions, which is not very neat when printing columns of dollars and cents. The following few lines inserted in a program will print the value of the variable, A, right-adjusted with the decimal point in the 25th (or any) column.

```
120 A$=STR$(INT(A))
125 IF INT(ABS(A))<1 THEN A$="1"
130 PRINT TAB (25-LEN(A$));A
```

also

```
10 A=INT(100xA+.5)/100
```

will round off A to dollars and cents.

P.S. We love our PET. James Standley, Hastings College, Hastings, Nebraska 68901.

SHARP TAPE DECK

In the Mar-Apr issue, page 55, TAPE TIPS suggests short tapes to find programs easily. The Sharp 1155 tape deck has an automatic program search system that looks for blank spaces between programs in either rewind or fast forward. By saving a computer program, then using the Sharp deck pause control to insert 10 secs of blank tape, the next program can be saved and any program can be found by counting the stops due to blank spaces. It takes only seconds to find the 9th or 10th program on a tape. Jack Clark, Oxton Hill, MD

PET-HAM INTERFACE

Ron Lodewyck, co-author of Commodore's Basic BASIC tutorial tape (a nice job, from what we've heard) is offering a PET-Ham interface called the M-65. It allows you to send and receive Morse code as well as radio teletype code; it is also a Morse code trainer. The hardware plus software package is available in kit form for \$69.96, or \$99.95 assembled. For more information, write Microtronics, 5943 Pioneer Road, Hughson, CA 95326. Or telephone (209) 634-8888.

MICROSIGNAL CATALOG

Another company offering PET peripherals and software is Microsignal, PO Box 161988, Sacramento, CA 95816 (see earlier issues of *People's Computers* for additional sources). Their catalog lists a voice input device, a sound generator, and a device that allows you to use a Teletype as a printer for the PET. Software for the most part exercises the peripherals offered by the company.



RS-232 PRINTER ADAPTER

Connecticut microComputer announces the first in a line of peripheral adapters for the Commodore PET. The PET ADAPTER model 1200 drives an RS-232 printer from the PET IEEE-488 bus. The PET ADA 1200 allows the PET owner to obtain hard copy program listings, and to type letters, manuscripts, mailing labels, tables of data, etc., using a standard RS-232 printer.

The PET ADA model 1200 is available assembled and tested, without power supplies, case, or RS-232 connector for \$98.50 (plus \$5.00 for shipping and handling) or complete for \$169 (plus \$5.00 for shipping and handling). Specify baud rate when ordering. (300 baud is supplied unless otherwise requested.) Contact: Connecticut microComputer, 150 Pocono Rd., Brookfield, Ct; (203) 775-9659

LIGHT PEN

A self-contained light pen which plugs directly into the Commodore PET 2001 user port has been announced by the 3G Company. This light pen makes it possible to bypass the PET's keyboard and interact directly with the information displayed on the CRT screen.



The light pen adds versatility to most graphics programs and is valuable as a teaching aid for young children. It also adds unique capabilities for application programs aimed at the non-computer oriented person. The light pen is complete and ready to plug into the PET. A sampled program and programming instructions comes with the pen. The entire package sells for \$24.95 and is available from 3G Company at Rt.3, Box 28A, Gaston, Oregon 97119; (503)985-7176

PET LISTING CONVENTIONS

PET Program listings in *People's Computers* employ the following conventions to represent characters that are difficult to print on a standard printer: Whenever square brackets appear in the listing, neither the brackets nor the text they enclose should be typed literally. Instead, the text between the brackets should be translated to keystrokes. For example, [CLR] means type the CLR key, [3 DOWN] means [DOWN, DOWN, DOWN] i.e., press the first CRSR key three times.

PET BLOOPERS

Woops! Several readers have informed us that the PLOT program by Philip Gash which we published in last issue's (V7, No.1) SPOT section will not run correctly as published. Indeed, lines 560 and 620 should be corrected as follows: (the underscore marks the corrected error):

```
560 X=D
620 INPUT "MAX VALUE OF Y(X)";Y4
```

POSTSCRIPT TO VIDEO MIXER

Last issue, in my article on the 'PET Video Mixer,' I mistakenly wrote that the three video signals to be mixed into a composite video output came from the PET IEEE 488-bus. This is NOT the case; the signals actually come from pinouts on the PARALLEL USER PORT. For references to the 'IEEE 488-bus' in the article and the schematic diagram, one should substitute PARALLEL USER PORT. Also note that there is a drawing error in one of the wires going to a pin on IC₁. The wire that goes from the 20k ohm trim pot should have been drawn connected to pin 7 of IC₁, not pin 6. Indeed, the circuit will not work properly if pin 6 instead of pin 7 is wired to +5 volts. Also omitted from the schematic is indication that the positive side of the 100 microfarad cap is connected to the 8-9V power supply through a 75 ohm resistor.

One fellow PET user who used this schematic to wire up his own video mixer pointed out to me recently that this circuit does not work with all monitors. This circuit, as is, will work only with monitors (or modified TV circuits) that require a POSITIVE video voltage in. If you need a negative video sync voltage for your particular monitor, then you could substitute a 7404 or 7416 IC for IC₃. 7407 is an inverter IC and is designed to produce a positive-going video signal. Besides determining whether you need a positive or negative-going out-put signal for the mixer to your monitor, you will also need to experiment with what voltage level works best with your particular monitor. If you have further questions or want more information about the Video Mixer, please contact me.

Randall Julin, 15 Poncetta Dr #322, Daly City, CA 94015 (415) 469-1157 (Day) (415) 922-6946 (Home)

HORSES

BY ANDY STADLER

This horse-race game was written by 12-year old Andy Stadler, of Albany, California. I met Andy at a local PET User's Group meeting; thanks for letting us share your program with People's Computers' readers, Andy.

HORSES lets from 1 to 9 players bet on one of 10 horses. Odds start at 10 to 1. If a horse wins, its odds go down by 1; if a horse loses, its odds increase by 1 (see lines 1140 and 1150).

As published, the 'track' for each horse

prints across the screen from left to right. If you prefer a faster track (heh heh, she said, in fair imitation of ye honorable editor Kahn) 500 PRINT "[RVS, 40 SPACE, OFF]";

One friend who tried out the game insisted upon whistling the traditional 'they're off and running' music once the horses were at the gate. To allow time for this, you may wish to insert a pause at the appropriate time by including the line:

605 FOR X=1 TO 500: NEXT X

in your program. — PC



HORSES

```

10 POKE 59468,14
20 REM ***HORSES***BY ANDY STADLER***
30 FOR X=1 TO 10: A(X)=10: NEXT X
40 PRINT "[CLR]"; SPC(10); "HORSES"
50 PRINT "[3 DOWN]HAVE ALL PLAYERS PLAYED BEFORE?";
60 GET A$: IF A$="" THEN 60
70 IF A$<>"Y" AND A$<>"N" THEN 60
80 PRINT CHR$(ASC(A$)+128)
90 IF A$="Y" THEN 230
100 PRINT "[UP] THIS IS THE GAME OF HORSES. IN IT,"
110 PRINT "[DOWN]YOU BET ON ONE OF THE 10 HORSES. EACH"
120 PRINT "[DOWN]STARTS WITH ODDS OF 10:1. IF A"
130 PRINT "[DOWN]HORSE WINS, ITS ODDS GO DOWN BY 1."
140 PRINT "[DOWN]IF A HORSE LOSES A RACE, ITS"
150 PRINT "[DOWN]ODDS GO UP BY 1. YOU START WITH 200"
160 PRINT "[DOWN]DOLLARS. EACH TIME YOU BET, THE "
170 PRINT "[DOWN]ONLY LIMIT IS YOUR BANKROLL!"
180 PRINT "[2 DOWN]DO YOU UNDERSTAND?"
190 GET A$: IF A$="" THEN 190
200 IF A$<>"N" THEN 220
210 PRINT "n": PRINT "THEN FIND SOMEONE WHO DOES,
YOU $8\#": GOTO 1120
220 PRINT A$
230 PRINT "[CLR]"; SPC(10); "HORSES"
240 PRINT "[3 DOWN]HOW MANY PLAYERS (1-9)? ";
250 GET B: IF B=0 THEN 250
260 PRINT B
270 PRINT "[DOWN]": FOR X=1 TO B
280 PRINT "PLAYER #";X;"NAME: ";: P$(X)="
290 PRINT "[6,LEFT]";
300 GET XX$: IF XX$="" THEN 300
310 IF ASC(XX$)=13 THEN PRINT "[2 SPACE]": GOTO 340
320 LET P$(X)=P$(X)+XX$: PRINT XX$;
330 GOTO 290
340 LET B(X)=200: NEXT X
350 PRINT "[CLR]"; SPC(10); "HORSES"
360 PRINT "[2 DOWN]THE HORSES ARE: [DOWN]"
370 RESTORE
380 FOR X=1 TO 10
390 READ A$
400 PRINT "#"; X; ": ";A$;TAB(25);"ODDS:";A(X);"TO 1"
410 NEXT X: PRINT "[DOWN]";
420 FOR X=1 TO B: IF B(X)<=0 THEN 470
430 PRINT P$(X);", # HORSE, BET":;INPUT C,D
440 IF C<1 OR C>10 OR D<0 OR D>B(X) THEN 460
450 LET Z(X)=C: LET Y(X)=D: GOTO 470
460 PRINT "[UP]";: GOTO 430
470 NEXT X
480 PRINT "[CLR, 2 DOWN]";
490 FOR X=1 TO 11
500 FOR Q=1 TO 40
510 PRINT "[RVS, SPACE, OFF]";
520 NEXT Q
530 PRINT "[DOWN]";
540 NEXT X
550 PRINT "[HOME]"
560 PRINT "[2 DOWN]";
570 FOR X=1 TO 10
580 LET C(X)=1
590 PRINT X; "[2 DOWN, 3 LEFT]";
600 NEXT X
610 PRINT "[HOME]";
620 REM***THEY'RE OFF!***
630 PRINT "THEY'RE OFF! [4 SPACE, HOME]";
640 LET W=INT(10*RND(1)+1):IF C(5)=5 THEN
PRINT "[HOME, 13 SPACE, [HOME]";
FOR X=1 TO (W*2)+1
650 PRINT "[DOWN]";
660 NEXT X
670 LET C(W)=C(W)+1
680 PRINT SPC(C(W)-1); W; "[HOME]";
690 IF C(W)<39 THEN 640
700 PRINT "THE WINNER:";RESTORE:FOR X=1 TO W
710 READ A$: NEXT X: PRINT A$;"(NO.":W;")";
720 FOR X=1 TO 5000:NEXT X
730 REM***COMPUTE WINS AND LOSSES**
740 PRINT "[CLR]";:FOR X=1 TO B: IF B(X)<=0 THEN 810
750 PRINT "[DOWN]";P$(X);",YOU ";
760 IF Z(X)=W THEN PRINT "WON";
770 IF Z(X)<>W THEN PRINT "LOST";
780 IF Z(X)=W THEN PRINT A(W)*Y(X);
790 IF Z(X)<>W THEN PRINT Y(X);
800 PRINT "DOLLARS."
810 NEXT X
820 FOR X=1 TO B:IF B(X)<=0 THEN 860
830 IF Z(X)=W THEN LET B(X)=B(X)+(A(W)*Y(X))
840 IF Z(X)<>W THEN LET B(X)=B(X)-Y(X)
850 IF B(X)<=0 THEN PRINT "[DOWN]"; P$(X);
" YOU BUSTED!"
860 NEXT X
870 FOR X=1 TO 5000:NEXT X
880 PRINT "[CLR, 6 SPACE]THE STANDINGS: [DOWN]";
LET Q=0
890 FOR X=1 TO B: IF B(X)<=0 THEN 920
900 PRINT P$(X); TAB(10); "$";B(X)
910 GOTO 930
920 Q=Q+1:GOTO 940
930 Z9=X
940 NEXT X: IF Q=B-1 AND B>1 GOTO 1110
950 IF Q=B THEN PRINT "YOU'RE ALL DEAD!": END
960 PRINT "[DOWN] ANOTHER ROUND? ";
970 GET A$: IF A$="" GOTO 970
980 IF A$<>"Y" AND A$<>"N" GOTO 970
990 IF A$="Y" GOTO 1130
1000 PRINT A$: GOTO 1170
1010 DATA "FIREBALL"
1020 DATA "SEATTLE SLOW"
1030 DATA "ZAPPING ZING"
1040 DATA "FRED'S FOLLY"
1050 DATA "WORRYSOME WART"
1060 DATA "HELL'S ANGEL"
1070 DATA "HEAVEN'S DEVIL"
1080 DATA "TELLY'S SON"
1090 DATA "WHAT A HOPE!"
1100 DATA "KEEP WISHING!"
1110 PRINT "[3 DOWN]"; P$(Z9);",YOU WON!!! WITH $";
B(Z9); "!!!"
1115 PRINT "JOLLY GOOD SHOW!!"
1120 GOTO 1170
1130 FOR X=1 TO 10
1140 IF X=W THEN A(X)=A(X)-1
1150 IF X<>W THEN A(X)=A(X)+1
1160 NEXT X: GOTO 350
1170 PRINT "[3 DOWN] COME AGAIN SOON!":END

```


M U J U M B L E

```

20 REM: PROGRAM NAME "JUMBLE"
30 REM: WRITTEN BY M.C. HOFHEINZ
40 REM: STOCKTON CALIF, MAY 1,1978
50 REM: ASSIGN A Z(Y) VALUE TO EACH LETTER
100 PRINT "TYPE ANY WORD OF 6 OR FEWER LETTERS"
150 INPUT N$
200 FOR Y=1 TO LEN (N$)
300 Z$(Y) = MID$(N$,Y,1)
400 NEXT Y
500 REM: START NESTED LOOPS TO INTERCHANGE LETTERS
600 FOR A=1 TO LEN(N$)
700 FOR B=1 TO LEN(N$)
710 REM: LEN(N$)TESTS FOR WORDS SHORTER THAN 6 LETTERS
715 IF LEN(N$)<3 THEN 1000
800 FOR C=1 TO LEN (N$)
805 IF LEN(N$)<4 THEN 1000
900 FOR D=1 TO LEN(N$)
905 IF LEN(N$)<5 THEN 1000
910 FOR E=1 TO LEN(N$)
915 IF LEN(N$)<6 THEN 1000
920 FOR G=1 TO LEN(N$)
990 REM: SORT SO THAT EACH LETTER IS PRINTED ONLY ONCE
1000 IF A=B THEN 2300
1025 IF LEN(N$)<3 THEN 2000
1100 IF A=C OR B=C THEN 2200
1225 IF LEN(N$)<4 THEN 2000
1250 IF A=D OR B=D OR C=D THEN 2100
1450 IF LEN(N$)<5 THEN 2000
1500 IF A=E OR B=E OR C=E OR D=E THEN 2070
1800 IF LEN(N$)<6 THEN 2000
1820 IF A=G OR B=G OR C=G OR D=G OR E=G THEN 2050
2000 PRINT Z$(A); Z$(B); Z$(C); Z$(D); Z$(E); Z$(G); " "; :IF LEN(N$)=5
    THEN PRINT " ";
2002 IF LEN(N$)=6 THEN PRINT " ";
2005 IF LEN(N$)<3 THEN 2300
2010 IF LEN(N$)<4 THEN 2200
2020 IF LEN(N$)<5 THEN 2100
2030 IF LEN(N$)<6 THEN 2070
2050 NEXT G
2070 NEXT E
2100 NEXT D
2200 NEXT C
2300 NEXT B
2400 NEXT A

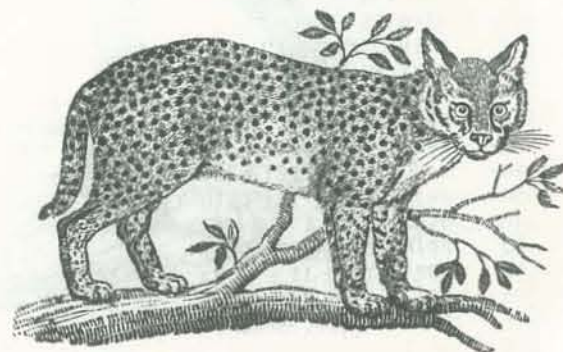
```

BY M.C. HOFHEINZ

M.C. Hofheinz' 'kids' programs are definitely for kids of all ages. In a previous issue, we published several short graphics programs; here's a word-oriented one. Note that in lines 2000 and 2002, various numbers of blanks are inserted to keep words from being printed partly on one line and partly on another. The program requires more than one screenfull to display all the combinations for 5- and 6-letter words. You may wish to study some of the combinations before they scroll out of sight: just press the STOP key. To continue, type the word CONT then press the return key. — PC

Another program for the kids. This one takes any word (or number) up to 6 characters and prints out every possible combination of those characters. The program was originally designed to help my kids solve the JUMBLE scrambled word puzzle which appears in the *San Francisco Chronicle*, and other papers.

A peculiarity of the PET makes it necessary to use A, B, C, D, E, and G (not F) since 'F OR' merges into FOR and produces a SYNTAX ERROR (see line 1820). This is due to the fact that FOR is an instruction word. □



BY THE DRAGON

The Dragon (alias Bob Albrecht) started this magazine in 1972. For its first four years, when it was an oversized, funky newspaper called PCC, he was also the editor. Recently, the Dragon has become interested in setting up and running fantasy games for children. *Dragonsmoke* is his forum for sharing information and ideas in this regard. —BK

The world of fantasy role-playing games is growing rapidly—perhaps as rapidly as the world of personal computing. And, slowly, the two worlds are blending, especially in the San Francisco Bay Area. We expect to see lots of computer activities at Pacific Encounters, a fantasy and science fiction games convention in San Mateo, CA on Labor Day weekend. We'll be there and will report on what happened.

In the meantime, here is an outstanding resource for you people (or other creatures) who want to get started in fantasy role playing adventure games.

RUNEQUEST

By Steve Perrin and Friends

Aha! Finally, a book a beginner can read and understand. This book is superb. It tells you what a fantasy role-playing game is, how to create an 'adventurer', spells out the mechanics of playing, tells much about magic—and lots more. Interlaced throughout the book are the sagas of Rurik the Restless, Ariella the Priestess and other adventurers. These sagas are specific examples of how the game might occur as it is played. Beautiful! Here are some excerpts:

WHAT IS A FANTASY ROLE-PLAYING GAME?

A role-playing game is a game of character development, simulating the process of personal development commonly called 'life.' The player acts a role in a fantasy environment, just as he might act a role as a character in a play. In fact, when played with just paper and pencil on the game board of the player's imagination, it has been called 'improvisational radio theatre.' If played with metal and plastic figurines, it becomes improvisational puppet theatre. However it is played, the primary purpose is to have fun.

SOCIOLOGICAL BASE

Giorontha is an Ancient Period and early Dark Ages world. It has far more to do with Mesopotamia, Ancient China, Hyboria, and Lankmar than it does with Medieval Europe, *Le Mort' D'Arthur*, or the Carolingian Cycle. Its heroes are Conans, Grey Mousers, and Rustums, not Lancelots, Percivals, and Rolands.

Unlike the worlds in other role playing games, there is no Alignment, as such. People have allegiances to nations, cities, religions, and tribes, not to abstract concepts. It is also possible for people within the game to survive quite well with no allegiances whatever except to themselves.

In *Giorontha*, the gods, in the forms of their followers and cults, play an active and important part in more major events.

However, most gods are complementary, and rarely oppose each other directly. Only the gods of Power are actively antagonistic, and even then only within their own spheres of interest.

PURPOSE OF THE GAME

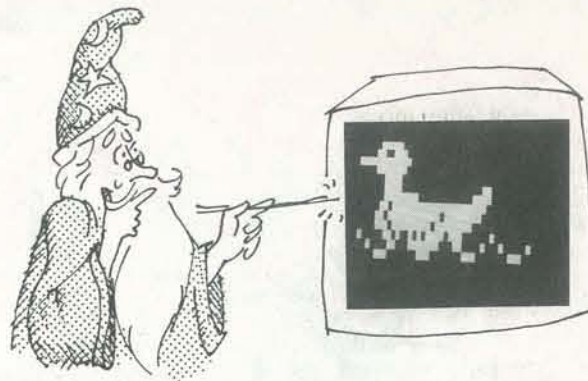
The title of the game, *RUNEQUEST*, describes its goal. The player creates one or more characters, known as Adventurers, and plays them in various scenarios designed by a Referee. The Adventurer has the use of combat, magic, and other skills to survive and gain glory, advancement in his skills, and treasure. The Referee has the use of assorted monsters, traps, and his own wicked imagination to keep the Adventurer from his goal within the rules of the games. A surviving Adventurer gains experience in fighting, magic, and other skills, as well as money to purchase further training.

The Adventurer progresses in this way until he is so proficient that he comes to the attention of the High Priests, Sages, and Gods. At this point he has the option to join a Rune Cult. Joining such a cult gives him many advantages, not the least of which is aid from the god of the cult.

Acquiring a Rune by joining such a cult is the goal of the game, for only in gathering a Rune may a character take the next step, up into the ranks of Hero, and perhaps Superhero.

Price: \$8. Available from: The Chaosium, PO Box 6302, Dept P, Albany, CA 94706.

THE SORCERER™ of EXIDY

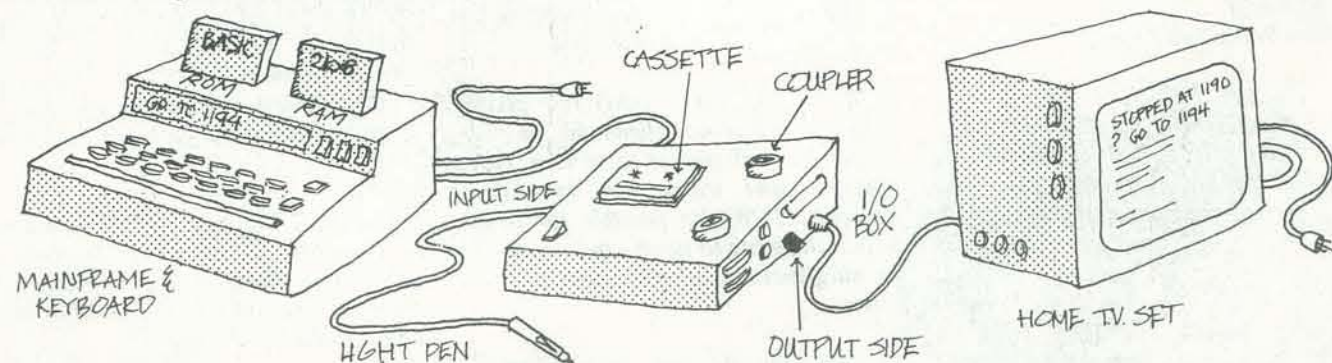


A REPORT ON AN EXCITING NEW PERSONAL COMPUTER
BY BOB KAHN

A BYTE Of History. Five years ago, in the September, 1973 (Vol 2, No 1) issue of this magazine—when it was still called *PCC*—Gregory Yob wrote an article entitled: 'Buy Your TV Set Something Nice, Like a Computer.' In that article, Yob presented a set of design and construction ideas regarding a consumer's computer that could be connected to a home TV set like video games. Yob, Bob Albrecht and others at *PCC* brainstormed these ideas while playing with a \$12,000 BASIC-speaking Hewlett-Packard 9830 Desk Top Computer.

One of the interesting features of the HP 9830 was its plug-in ROM cartridges which allowed the user to add embellishments such as matrix operations and strings to the machine's standard BASIC. Knowing that ROMs would get much cheaper and that eventually they would contain entire languages, Yob et al. came up with the notion of developing cartridge pack ROMs and RAMs which could be plugged into the mainframe of a home computer. The article listed several possible ROM cartridges including programming languages (BASIC, PILOT...), games, home applications and a few other esoteric odds and ends. Finally, Yob hopefully predicted that a minimum unit (keyboard + mainframe with 16K RAM cartridge and a 'starter set' of ROMs) should cost about \$500.

PCC published this article more than a year before MITS announced the ALTAIR™. Four years later several 'consumer's home computers' did, indeed, appear on the market—



An artist's depiction of a personal computer which connects to a home TV set. (From *PCC*, Vol. 2, No. 1, Sept. 1973, p. 5). Notice the resemblance to an HP 9830 Desk-top Computer with plug-in ROM and RAM cartridges.

first from Commodore, and shortly thereafter from Radio Shack, with other manufacturers preparing in the wings.

Abacadabra . . . Out of the Hat Comes . . . Exidy's Sorcerer! Eagerly watching this consumer market from his excellent vantage point as founder of BYTE Computer Stores (now BYTE Industries) was Paul Terrell (see *PC*, Vol. 6, No. 5, pp. 28-31). With the acumen of a true wizard, Terrell closely monitored the successes and failures of various personal computers. In particular, he zeroed in on the PET, TRS-80 and SOL, paying close attention to graphics and adaptability features. Closely paralleling the 1973 *PCC* notion, Terrell started formulating the features and specifications of a computer to be priced on the 'consumer range' but that could also satisfy the needs of hobbyists, businessmen and computer professionals as well as non-technically oriented consumers.

To actualize his 'dream machine', Terrell, (having sold BYTE Industries to Logical Machines) joined forces with Exidy, Inc. in November, 1977. Behind Bally and Atari, Exidy is the third largest manufacturer of coin operated video games in the country. It was founded by H.R. 'Pete' Kauffman, a former principal of Ramtek Corp., manufacturer of high resolution graphic systems. Exidy is privately owned and located in Sunnyvale, California—in the heart of Silicon Valley.

Exidy formed a Data Products (i.e. computer) Division, with Terrell as marketing manager, which set out early in 1978 to

design and produce Terrell's 'dream machine'. Howell Ivy, another former Ramtek employee—now Exidy vice president of Engineering, designed and engineered the computer last winter and spring. Exidy dubbed its computer 'The Sorcerer™', and announced it early in June—just in time for the National Computer Conference in Anaheim. Indeed, with a prototype Sorcerer and a breadboarded color version of the machine on display in the Personal Computing Section of NCC, Exidy conjured up considerable interest.



The Exidy Sorcerer™ with BASICROM Pac™ protruding on the right side of the machine.

The Sorcerer's Magic. Looking at the Sorcerer from a distance or in a photograph, it resembles a number of currently available micro-computer systems such as the Apple II, the TRS-80 and particularly, the SOL; it is basically a typewriter-style ASCII keyboard with separate 16 key numeric pad mounted in a molded case. When plugged into a video display (not included) and a cassette recorder (also not included), the Sorcerer becomes a fully functioning computer system. But even at a distance, the Sorcerer has one immediately noticeable feature that distinguishes it from every other micro computer system currently on the market: plugged into the right side of the machine is the Sorcerer's 16K ROM Pac or or plug-in cartridge ROM. Just as proposed in the 1973 *PCC* article, this plug-in ROM Pac allows the user to easily change programming languages or to plug in specific applications such as a text editor. The Sorcerer comes with a Standard BASIC ROM Pac containing version 4.52 of Microsoft BASIC. ROM Pacs for other languages (such as Z-80 assembly language, APL, PILOT, FORTRAN and COBOL) and for specific applications such as word processing are currently under development; an EPROM Pac which allows the user to create his own plug-in application is also available. In addition to the 16K ROM Pac, the Sorcerer has 4K internal ROM containing an operating system with a power-on monitor program and 8K of RAM for user program space; the 8K RAM is internally expandable to 32K.

Looking more closely at the keyboard, the Sorcerer has even more magical powers. The keyboard provides a full set of 128 standard (upper and lower case) ASCII characters as well as a set of 64 graphics characters (much like those on the Commodore PET) and yet another set of 64 characters that may be defined by the user. The Sorcerer can display a total of 1920 characters on the screen at one time in 64 characters

by 30 lines in an 8 by 8 format. This works out to 512 by 240 points which may be addressed in graphics mode.

The Sorcerer comes with a dual cassette interface with software selectable data rates of 300 or 1200 BAUD. It will accept cassette tapes directly from a SOL-20 and from General Recording Tape (GRT) Corporation's G-2 library of BASIC applications programs—among others.

The Sorcerer is based on the Z-80 microprocessor. It has a parallel I/O interface which is directly compatible with a Centronics line printer; it also has an RS-232 serial interface and thus may be used as an intelligent terminal connected to a host computer via MODEM and telephone at either 300 or 1200 BAUD. The Sorcerer is compatible with the S-100 bus; a 6-slot S-100 expansion unit may be added to the basic computer. The Sorcerer is directly compatible with a Hitachi TV/Monitor or with Exidy's high resolution video monitor. Up to two cassette recorders may be connected.

Looking Up the Sorcerer's Sleeve. The Sorcerer comes completely assembled and factory tested and is priced at \$895 including the Standard BASIC ROM Pac and 8K user RAM—but without video monitor or cassette recorder. The high resolution 12" monitor from Exidy is listed at \$299, and cassette recorders can run from \$30 to +300+. Thus, if you don't happen to have an appropriate TV or cassette recorder around, a minimum Exidy computer system will cost between \$1100 and \$1400, putting the Sorcerer more in the price range of an Apple II than a PET or TRS-80.

The Sorcerer comes with an operation manual and a BASIC programming manual. Exidy plans to market the Sorcerer through computer stores across the country. The standard warranty on the machine is 90 days (parts and labor), and machines are expected to be serviced through the stores where they were purchased.

Exidy is currently busily designing a 256 by 256 eight color version of the system which will require the S-100 expansion unit and two S-100 boards for operation. It will have a standard video output signal so that it can be plugged in directly to a color TV/monitor. The company is also designing a floppy disk-based system which will be available by the end of this year.

As it stands, the Sorcerer appears to be the most well-designed personal computer currently on the market in its price range. It is not exactly a 'consumer item' at 150% the cost of a TRS-80 or a PET—nor in terms of its rather limited marketing and distribution set up. But it certainly will be a strong contender in the personal computer marketplace, and we look forward to trying one out and bringing readers of *People's Computers* a first hand report on its magical powers. □



SORCERER and ROM Pac are registered trademarks of Exidy, Inc.

ANNOUNCEMENTS

HARDWARE



**COMPUCOLOR II —
Personal Computing in Color**

CompuColor Corporation announces the latest in personal computers—the CompuColor II, or 'The Renaissance Machine'.

The CompuColor II, available in 5 models, will have its own 8-color, 13-inch diagonal display, a typewriter-like keyboard with 3-key rollover, 8080A CPU, 4K to 32K Memory (depending on the model), and the only built-in high reliability mini-disk drive mass storage device available on a home computer. Moreover, CompuColor II supports BASIC 8001.

CompuColor has developed a complete games library on diskettes designed to take advantage of the CompuColor II's advanced color graphics hardware. For example, StarTrek, Blackjack, Chess, Checkers, Othello, Biorhythms, and educational games like Math Tutor are available. However, the new CompuColor II does more than play games; it is a full-fledged microcomputer system with the power to handle complex tasks—home budgeting, for instance. Programs for checkbook balancing and income tax compilation are available

right now. Load your records from the CompuColor II to a diskette and you've got a permanent record of all your personal business transactions.

Prices start at \$795. A full-blown CompuColor II graphic system with 32K is priced at \$2395. For further information please contact: Joy Baker, CompuColor Corporation, 5965 Peachtree Corners East, Norcross, Georgia 30071. (404) 449-5961.

TERAK COMPUTER

TERAK is a complete, stand-alone, microcomputer system based on DEC's LSI-11. It includes 56K Bytes of RAM, single floppy disk, serial interface, video electronics, 12" CRT and keyboard. It is designed as a graphics system using a 320 X 240 bit map. Text is displayed 24 lines X 80 characters. Text and graphics can be displayed simultaneously. A complete disk operating system is available including single and multi-user BASIC, FORTRAN IV, APL, PASCAL. Hardware is priced at \$7850, software is priced separately. Contact TERAK Corporation, 14425 N Scottsdale Rd Suite 100, Scottsdale, AZ 85260.

PORTABLE MICROCOMPUTER

Adaptive Systems, Inc. announces a battery operated microcomputer, Model 6000. It is a PDP-8 class of computer using an IM 6100 microprocessor, which executes the PDP-8/E instruction set. It runs up to 8 hours battery operating time on one charge. For information contact: Adaptive Systems, Inc, PO Box 1481, Pompano Beach, FL 33061.

NIBBLER

The 'Nibbler' is a low cost home/hobby computer. It's easy to use because it

includes a Basic interpreter in ROM on the board. The Nibbler is also an easily programmed intelligent controller for industrial applications.

The Nibbler, a single board microcomputer, is built around the National Semiconductor SC/MP-II microprocessor and NIBL, a 4K Basic interpreter, in read only memory. 2K bytes of random access memory are provided for user programs, allowing programs of up to approximately 60 average statements to be run.

The Nibbler is not a kit; it's an assembled and tested finished product.

The Nibbler is priced at \$149.95, and a detailed hardware/software manual for the Nibbler is available from Digi-Key for \$5.00. Contact: Digi-Key Corporation, PO Box 677, Thief River Falls, MN 56701 (218) 681-6674.



VERSATILE 3B OR VERSATILE 4: FULL OPERATING SYSTEM IN A SINGLE UNIT

Computer Data Systems of Wilmington, Delaware has announced availability of their disk based computer systems, the VERSATILE 3B and their expanded version, the VERSATILE 4. This single unit computer combines a 9" video screen with 24x80 display, built-in mini-floppy disk drive with 143K bytes of storage, upper/lower case alphanumeric keyboard, separate numeric keypad and all electronics within a single durable plastic enclosure. The computer

mainframe incorporates the 8085 CPU, 24K static RAM and a serial I/O port with RS-233 connector. The VERSATILE 4 expands on this system by providing 32K static RAM and 315K bytes of storage. Operating software is supplied with both units and includes 20K Extended BASIC by Micropolis, a Disk Operating System and a complete software library of demonstration programs.

Five diskettes comprise the Software Library. The first contains the Disk Operating System and 20K Extended BASIC. Multi-statement lines and multi-dimension arrays are incorporated. The second diskette contains several games including LUNAR, HORSE RACE and BLACKJACK with room left for users to add their own. Disk #3 contains a Small Business Accounting Package which includes programs for financial analysis, mailing list, inventory, and more. Disks #4 and #5 are blank so systems users may enter their own programs. Contact: Computer Data Systems, 5460 Fairmont Drive, Wilmington, DE 19808. (302) 738-0933.

SOFTWARE

TRS-80 PROGRAMS

Farrell Enterprises has announced the availability of the following programs for the TRS-80:

- **MAGAZINE CATALOGING PROGRAM:** (Level II only) — \$7.98
This program allows you to catalog your library of magazines and search your library by magazine title, issue, article name, subject matter, author and certain combinations of these parameters. The program is easily modified for disk systems.
- **ELECTRONIC DESK CALENDAR** (Program is free is ordered with the Magazine Cataloging Program) (Level II only) — \$7.98. This program calculates the day of the week for any date from 1582 on; also allows you to write yourself a note or reminder which will be flashed on the screen on a particular date you specify.

- **STARTREK 80** (Level I, 16K and Level II) — \$7.98,
This program contains all the standard STAR TREK features including short and long range scans, warp engines, photon torpedoes and phasers. It also includes an experimental death ray, self-destruct, damage control, supernovas and random events that may take 6 weeks of play to discover.

- **DUNGEON ADVENTURE** (Level II, 16K only) — \$9.95
This program is too large to fit on one cassette. It allows you to descend into a dungeon in search of valuable treasure guarded by incredible monsters—the deeper you go, the bigger the treasure... and the more fearsome the monsters. Special programming techniques allow for later add-ons such as different dungeons, new monsters and more magic spells.

Address orders (California residents include 6% sales tax) and inquiries to Farrell Enterprises, P.O. Box 4392, Walnut Creek, CA 94596

TRS-80 LEVEL II SOFTWARE

The Microware Division of Physical Biological Sciences, Ltd. has customized two well-tested programs, 1) Mailing List and 2) Inventory Management, for use on the TRS-80 Level II system. Each program costs \$8.95 and comes packaged as a cassette with a 36 page user booklet. These programs are available at local computer stores or from MICROWARE DIVISION — PBS, Ltd., P.O. 47, Blacksburg, VA 24060.

TIME-SHARE FOR NORTH STAR

A Time-Share Disk BASIC System is now available for users of the North Star FloppyDisk System. Designed to operate with either 8080 or Z-80 processors, NORTHSHARE™ provides up to four independent users with selectable memory partitions and buffered terminal outputs.

Minimum memory requirements for operation are 24K bytes. There are no special hardware requirements outside of additional terminals and I/O ports to support the multiple users.

System includes one Diskette and Release 3 North Star Basic and DOS with NORTHSHARE™ Supervisor and Documentation Package. Price is \$48. Byte Shop of Westminster, 14300 Beach Blvd., Westminster, CA 92683; (714) 894-9131.

PILOT ON NORTHSTAR DISK

Through the kind cooperation of the author, Dr. John Starkweather, the source code for 8080 PILOT is now available in machine readable form from the North Star Software Exchange. The price is \$9.50 including the cost of the diskette. The diskette also includes a remarks file and an executable file containing a preliminary version of PILOT modified to run with the North Star DOS. This file also includes the standard PILOT demonstration program.

Anyone interested may order diskette NSSE5 from:
North Star Software Exchange
2547 Ninth Street
Berkeley, CA 94710

MICROSOFT'S COBOL-80

Microsoft has announced the first COBOL for 8080/Z-80/8085 microprocessor systems. COBOL-80 conforms to the 1974 ANSI standard, thus giving users immediate access to programs already written in COBOL. All Level 1 features and the most useful Level 2 options for the 'Nucleus' and for Sequential, Relative and Indexed file handling facilities are included. Additionally, Level 1 Table Handling, Library and Inter-Program Communication facilities are provided. Of the advanced Level 2 features, Microsoft has included the verbs STRING, UNSTRING, COMPUTE, SEARCH and PERFORM (varying/until), along with condition specification by way of condition-names, compound conditions and abbreviated conditions. COBOL-80 allows a packed decimal data representation to conserve memory on floppy disks.

The COBOL-80 system consists of two packages: a compiler for translating source code into relocatable object code, and a runtime system containing standard

routines needed by the object code at execution time. The whole system may be run in less than 32K bytes. Rate of compilation is 250 lines per minute. Complete documentation is supplied with the system or may be purchased separately for \$20. COBOL-80 is available off the shelf to run under the CP/M and ISIS-11 operating systems for \$750 per copy. Microsoft, 300 San Mateo, NE, Suite 819, Albuquerque, NM 87108; (505) 262-1486.

8080 TEXT PROCESSING SYSTEM

Technical Systems Consultants, Inc. announces availability of its 8080 TEXT PROCESSING SYSTEM which allows the use of over 50 commands for special text formatting applications. The commands included will support multiple spacing, left margin control, indenting, the ability to save contiguous text, paging, left hand justification, right hand only justification, left and right justification, centering, no-fill modes, page numbering, printing of left, right, or centered titles, and line length control. The TSC TEXT PROCESSING SYSTEM will also output numbers in either Arabic, capital Roman numerals, or small Roman numerals. Tab columns may be defined as well as the tab character and tab fill character.

An external editor is required as no editing functions are included. The TSC Text Processor resides in just over 8K beginning at 1000 hex plus filespace. The full manual including an 'Introduction to Text Processing', user's guide, and fully commented assembled source listing is priced at \$32.00. An Intel ASCII format paper tape is available for an additional \$9.00. Contact Technical Systems Consultants, Inc., Box 2574, West Lafayette, Indiana 47906.



OTHER

PERSONAL COMPUTER NEWS

The Personal Computer News is a monthly newsletter edited by Donald L. Wallace, and is published from Dayton, Ohio. PCN is dedicated entirely to a variety of reader services, and contains no paid advertisements. PCN features a regular news column detailing developments in the micro-computer industry and related technologies; objective product and software evaluations geared to the small businessman and the hobbyist alike; and two features due to commence this summer include a PCN-operational Software Exchange and a Trading Post classified advertisement section. In addition, a Software Sources listing culls the latest entrepreneurial offerings from the microcomputer media, and an Index to Computer-Related Articles cross-references features in the popular computerist magazines. Subscription Rates: \$9/year in the US; \$15/year in Canada and Mexico; and \$24/year overseas. Personal Computer News, PO Box 425, Dayton, Ohio 45419

THE RECREATIONAL PROGRAMMER™

The *Recreational Programmer* is a new magazine devoted to leisure uses for computers and programmable calculators. It is based on the column of the same name that ran in the *Journal of the HP-65 User's Club*. The *Recreational Programmer* is an international forum for sharing programming ideas and stimulating diversions; it is geared to the interests of its readers and is not aimed at any one type of machine or logic system—AOS, RPN, Tiny BASIC, PILOT or FORTRAN can all be used to share ideas. Programs published in the *Recreational Programmer* will be accompanied by a written explanation in clear, logical language so that they may be adapted to any system.

The *Recreational Programmer* will be available bi-monthly by subscription for \$12 per year (\$15 foreign) with the charter issue starting in September,

1978. Subscriptions, correspondence and programs should be sent to the Recreational Programmer, Box 2571, Kalamazoo, MI 49003

SPHERE USERS NEWSLETTER

The Sphere Microcomputer User's Newsletter will be starting its third year of publication. The newsletter is mailed six times a year. It contains Hardware and Software features of primary interest to Sphere Microcomputer owners but also of great interest to any M6800 computer owner. Subscribers should remit \$12.00 domestic or \$16.00 Foreign to Programma Consultants, P.O. Box 70127 Los Angeles, CA 90070. Co-editors are: Roger J. Spott and Jeffrey Brownstein, 13975 Connecticut Avenue, Wheaton, MD 20906.

RCA COSMAC MAGAZINE

A new magazine and club to support the RCA 1802 COSMAC is QUESTDATA. Owners of Elf, Super Elf, Elf II, COSMAC VIP, COSMAC Development System or Homebrew 1802 will find many programs, applications, and experiments for their microcomputer in each issue of QUESTDATA.

QUESTDATA will be showing the complete RCA instruction set and how to build interesting programs for: graphics, control, games, and business purposes. Coverage will be given to Tiny BASIC, Elf Expansion Possibilities (memory, cassette I/O, etc.), light pens, reader questions and music programs.

QUESTDATA is published monthly. Annual subscriptions are \$12, available from QUESTDATA, P.O. Box 4430, Santa Clara, CA 95054. Foreign subscriptions, with the exception of Canada and Mexico, are \$6 extra for mailing.

RADIO SHACK COMPUTER USER NOTES

This non-profit, no advertising user's newsletter is published monthly and

serves both neophyte and experienced computer user. Published independently of Tandy Corp., these user notes disseminate news of interest to TRS-80 users, whether it is from Tandy, other vendors or from computer users. The publishers have experience with other similar newsletters. The subscription price is \$10 for 12 issues in the US (\$18 overseas), payable to Bookmakers, Box 158, San Luis Rey, CA 92068.

TRS-80 BULLETIN AND SOFTWARE

Please consider this a response to your 'Call for Distributors'. SoftSell Unlimited is a new entry in the micro-processor software arena, currently featuring software for the Radio Shack TRS-80 exclusively.

SoftSell has a twofold objective. The first objective is to provide TRS-80 owners practical and profitable applications for business and education. The second objective is to provide a newsletter to help the neophyte programmer and the advanced hobbyist know more about his TRS-80. This is accomplished through our 'SOFTSELL BULLETIN'. It will initially be a quarterly offering and the annual subscription fee is four dollars.

My real personal interest, though, is in educational aspects of micro-processors. I am currently developing a reading comprehension series for grades three to five and a very basic introduction to computers in the form of a video and audio approach to operating the TRS-80 for primary students. We are looking for programs in the educational field, and we would be glad to list them in our program library. We are always soliciting authors for programs (but please spare us the common games). I am interested in those games that will educate, particularly games that pit the operator against the computer. Ronald D. Stange, SoftSell Unlimited, P.O. Box 145, Lithonia, GA 30058.

PET WORKBOOK

A workbook, 'Getting Started with Your PET' is now available to PET users who

are anxious to put their PETs to work, but have suffered from a lack of documentation. The beginners workbook supplements the documentation provided by Commodore, covering the fundamentals of PET BASIC. The descriptive text is heavily laced with step-by-step, detailed exercises including the expected PET response.

Other workbooks covering advanced topics are also available. These topics include string handling, arrays and looping, graphics, cursor control, PEEK and POKE, Memory, programmed cassette I/O, real time clock, and linkage to assembly language subroutines.

The Company also provides PET software on cassettes. Contact: Total Information Services, P.O. Box 921, Los Alamos, NM 87544.

PET NEWSLETTER

TRANSACTION, a bi-monthly publication, is a new newsletter for owners of the Commodore PET. Contact: TRANSACTION, P.O. Box 461, Philipsburg, PA 16806.

5 MINUTE CASSETTES AND CASSETTE DUPLICATION

Microsette Company, manufacturer of premium quality audio cassettes announces the availability of custom made 50 foot (C-10) cassettes and cassette duplication services for the TRS-80, PET and other personal/home computers. The 50 foot cassettes run approximately 5 minutes per side (significantly reducing rewind time) and are priced as follows from Microsette:

- Sample (1) — \$1.00
 - 10-pack — \$7.50
 - 100-pack — \$58.00
- (Californians add 6% sales tax)

These cassettes are also available in local computer stores.

Microsette will duplicate cassette programs in any volume from 100 to 100,000 and guarantee each cassette for 60 days. Prices start at \$1.00 each which includes the cassette, box, unaffixed blank labels and shipping anywhere in the US.

Contact Microsette Co., 777 Palomar Ave., Sunnyvale, CA 94086.

NUTRIVALUE™

This practical home application of computers enables users to analyze recipes, meal plans, and daily or weekly menus for their nutritional content.

NUTRIVALUE I is written in BASIC without string variables and includes Nutrient data for 53 ingredients. NUTRIVALUE II utilizes string functions and stores nutrient data in a file external to the program.

Both programs are available with a data base of 100 or 200 ingredients. The Level I program is available in listing for \$10., on paper tape with list for \$13. The Level II listing is \$30; a tape or cassette is \$35 (add \$10 for 200 ingredients). NUTRIVALUE is available at your local computer store or from Consultus, P.O. Box 86, Arlington, MA 02174.

CALCULATOR USER'S CLUB

A number of readers of the newsletter CALCULATOR LIB have organized The Liberated Calculator-user's Club, a truly universal, independent non-profit group of calculator users (regardless of the make of the calculator,) dedicated to exploring the limits of the state of the art of calculator-mathematics. The overall goal is to profit mutually from all members' knowledge of calculators and related fields, and create a forum that allows club members to meet and identify with each other's interests. At this moment the Club needs volunteer members to act as officers in the editorial committee, correspondents, reporters and translators.

Presently, Club members speak/write English, French, German and Hungarian. This list will be hopefully expanded with readers in other countries. For more information send a large, self-addressed, stamped envelope to: Gene Hegedus, P.O. Box 2151, Oxnard, CA 93034. (805) 486-7191.