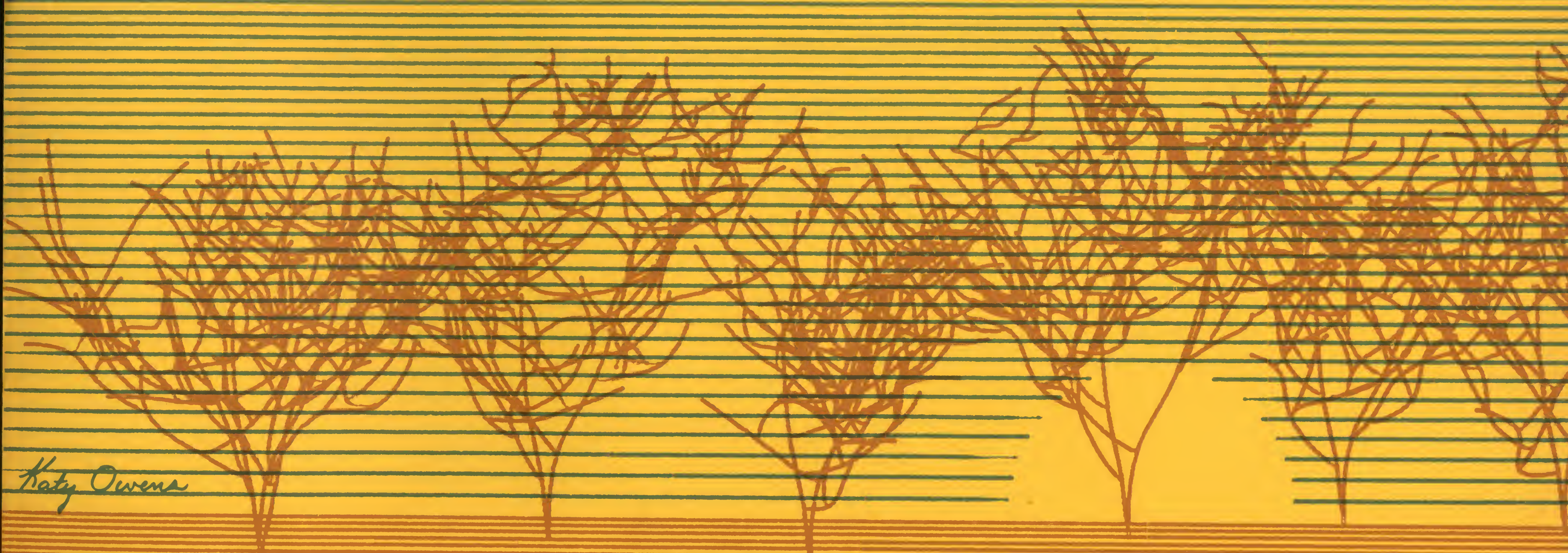
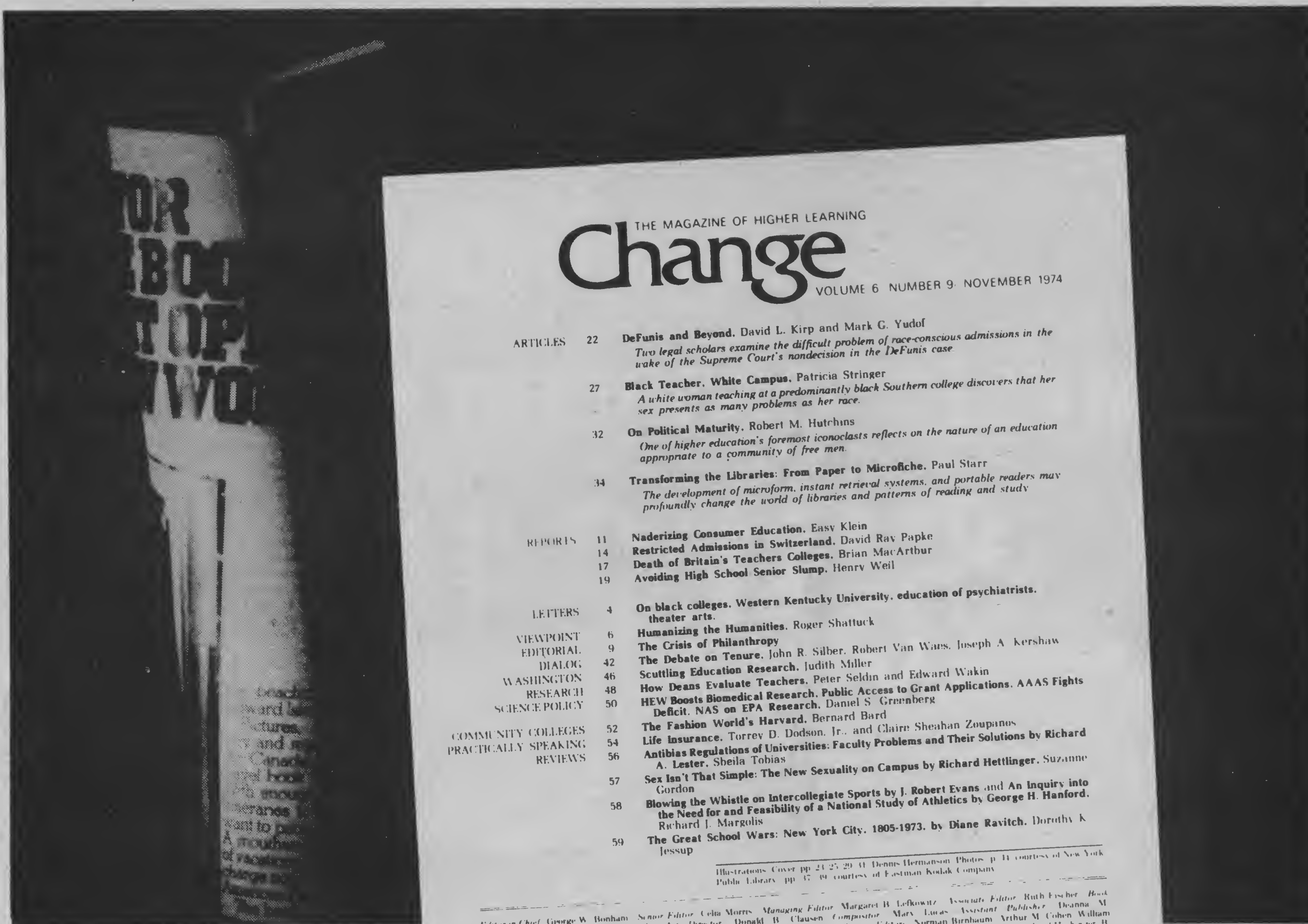
creative computing

a non-profit magazine of educational and recreational computing   **Mar-Apr 1975**   **$1.50**



Computer Careers   Games   Review of 34 Books on BASIC



Will Change Become a Habit? The Proof is in the Reading.

In the past few years, **Change** has become the leading platform of national debate over the future of higher learning. This is no small social issue, and if you are not already a member of this important group of **Change** readers, we hope that you will become one now.

The reasons for joining may be self-evident: Each month, **Change** brings you a view of the world of education not available anywhere else (as you can see by the sample contents page above). **Change** is written by some of the most perceptive thinkers in the country (and, incidentally, **for** the best minds in the country as well).

Each monthly issue of **Change** brings you exciting and provocative intellectual fare. Its stimulating articles are among the most widely quoted in the country. Its briefer **Change Reports** bring you exclusive, analytical journalism from the four corners of the world (average reading time: 5 to 8 minutes each). The regular columns in **Change** include Academic Research, Science Policy, Washington—guest editorials, some of the best book review essays around and a special section on community colleges. **Change** editorials are widely quoted and debated.

If you would like to test these claims, you can now do so without risking a penny. Here is a special offer: **simply mail in the coupon.** You will receive next month's copy of **Change**, and we will enter a one-year trial subscription for **$12** instead of the usual **$14**. If **Change** does not live up to your expectations, mark your invoice "cancel." No questions asked. But one small warning: reading **Change may be dangerous to some of your fondest misconceptions.**

CHANGE, NBW Tower
New Rochelle, NY 10801

Please enter my one-year no-risk enrollment to CHANGE at the special introductory rate of $12, saving me $2 over the current price. If I am not completely satisfied, I retain the right to cancel my subscription upon receipt of the first invoice, and will owe nothing.

☐ Bill me for $12
☐ Charge $12 to my credit card as indicated below.
☐ My $12 is enclosed. (If I should cancel, I will receive a full refund.)

Please charge my: P 6035

☐ BankAmericard
☐ American Express
☐ Master Charge

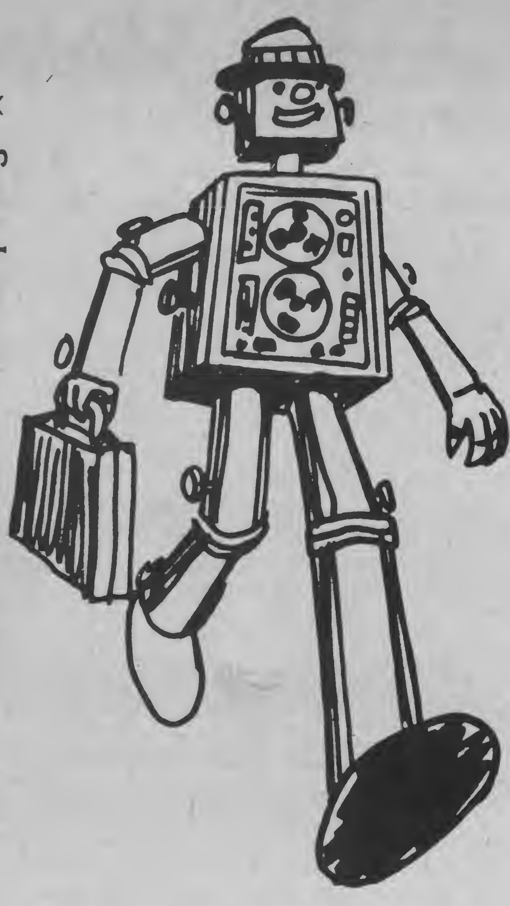
credit card information

____

Signature: ________________
Name ________________
Address ________________
City ________________
State ________________ Zip ________

Creative Computing

★ a non-profit magazine of educational and recreational computing ★



In This Issue of CREATIVE COMPUTING

COMPUTER CAREERS

ARTICLES

ACTIVITIES & GAMES

FOOLISHNESS

DEPARTMENTS

COMPUTER CAREER PHOTOS

The special focus of this issue is computer related careers. We're thinking of careers not only in traditional data processing but in many other areas which utilize a computer. Areas like medicine, industrial production, retail stores, weather forecasting, police protection, and air traffic control. Throughout the issue you'll find short photo stories that illustrate these and many other careers in the exciting, interesting world of computers.

THE COVER

The cover design, "The Dawn of Creative Computing" is an original work by Katy Owens. It was programmed in FORTRAN on an IBM 360/50 and was produced off line on a CalComp 563 31" plotter.

Katy has a BS in Computer Science from Ball State University, Muncie, Indiana where she now works as a systems programmer. She started in computer art less than a year ago with some original encouragement and assistance from Tom Huston of Computra, Inc.

Volume 1 Number 3

CREATIVE COMPUTING is published bi-monthly by Ideametrics, P. O. Box 789-M, Morristown, NJ 07960. Third class postage paid at Morristown, NJ. Subscriptions: Institutional 1-year $15, 3-year $40; Individual 1-year $8, 3-year $21; Student 1-year $6, 3-year $16. Foreign postage $1.50 per year additional. Single copies $1.50.

Earn Extra Money in Your Spare Time

READERS! We need more subscribers so we can financially break even, and then increase the size of *Creative Computing*. We can advertise but that costs more than it brings in. We can buy mailing lists and send out flyers, but good lists are hard to come by. There are other approaches, but the cost is generally higher than the payoff.

So we're coming to you readers, because we feel that your friends are the most likely new subscribers. We know you love us and you'd be happy to work for us for nothing, however, we're not asking that. We're going to pay you for your efforts. Here's the deal:

1. We'll pay you a 15% commission on the subscriptions you bring in.
2. Minimum of five subscriptions per order.
3. Payment must be enclosed with order (we can't possibly afford to send out bills on this deal.)

For example, say you get the following six subscriptions:

Type	*Cost*	*15% Commission*
1 Year Student	$ 6.00	$ .90
1 Year Individual	8.00	1.20
1 Year Library	15.00	2.25
3 Year Student	16.00	2.40
3 Year Individual	21.00	3.15
3 Year Library	40.00	6.00
TOTAL	$106.00	$15.90

You would send us six completed subscription blanks (or reasonable facsimilies), $106 in cash, check, or money order and your name and address. By return mail, you'll receive our check for $15.90.

Remember: don't send in fewer than 5 subscriptions, do enclose payment with the order, and do include your name and address. (One of the five subscriptions may be your own — new, renewal, or extension.)

CHANGE OF ADDRESS

Because of our new-fangled computerized system for keeping names and addresses of subscribers, we must have your OLD and NEW address in case of any change. Your CG number which appears above your name on the address label would help enormously too.

Zip Code. It is VITAL. All our files are kep in zip code order. If it's not on the label, your magazine could wind up destroyed (yes, destroyed, not returned to us) rather than delivered to you. Uncle Sam only has an obligation to deliver 3rd class mail if it has a zip code on the address. Don't forget ZIP!

CREATIVE COMPUTING SUBSCRIPTION FORM

Published bi-monthly —— 6 issues per year

INSTITUTIONAL
- ☐ 1-year $15
- ☐ 3-year $40

INDIVIDUAL REGULAR
- ☐ 1-year $ 8
- ☐ 3-year $21

INDIVIDUAL STUDENT
- ☐ 1-year $ 6
- ☐ 3-year $16

FOREIGN POSTAGE
- ☐ $1.50 per year

BILLING INSTRUCTIONS
- ☐ Payment enclosed
- ☐ Please bill me (Receive 1 issue less)

Name ______________________ Title or Dept. ______________________

School or Company ____

Street Address ____

City ______________________ State ______________ Zip ____________

Return form to: CREATIVE COMPUTING, P.O. Box 789-M, Morristown, New Jersey 07960

Input/Output

Dear People:
Please send me *Creative Computing* for one year. Enclosed is a check for $8.00. I've been looking for this kind of periodical for quite some time. Glad I finally found you.

George Pavel
Lawrence Livermore Lab

Dear Editor:
Thanks for sending along a copy of Volume 1, Number 1 of *Creative Computing*. It looks like a winner and I wish you and *Creative Computing* the very best of good fortune.

Richard B. Morron
Program Manager–DP Security, IBM

Dear Editor:
Congratulations on the first issue of *Creative Computing*! It should enjoy high acceptance among classroom instructors and others interested in computers in education. Best wishes for 1975.

Patricia A. Conway
Control Data Corporation

Dear Editor:
Only today did I have a chance to browse thru your *Creative Computing* issues.

My special interest in "Computer Mathematics", as you know, is the use of computer programming – in any language – to motivate students to search for areas of pure and/or applied mathematics which can present problems which need computers.

95% of the issue is aimed elsewhere – so from my point of view your magazine lacks creativity in mathematics.

Keep me informed.

George Grossman
Director of Mathematics
Board of Education of the City of New York

Dear Mr. Ahl:
Just a note to thank you for your contribution to this morning's Education Section. ["Spur to the Imagination," *The New York Times*, Jan. 15, 1975] The story and the box made a fine combination and I'm sure that many people found it fascinating. We appreciate your help, and needless to say, we would be happy to hear from you any time that you think there are some newsworthy events in the computer field that we should be aware of.

Edward B. Fiske
Education Editor
The New York Times

Dear Editor:
Please note in your publication the details of the O.A.M.E. conference, May 23-25, Univ. of Western Ontario, Canada. Part of the program will deal with computer education.

Rick Stephenson
Sir Wilfrid Laurier S. S.
450 Millbank Drive
London, Ontario, Canada

For the most part *Creative Computing* will not announce conferences or publish calendars of events. We feel that many other periodicals perform this role quite adequately. We will, however, report on significant conference sessions and meetings. Readers: if you have other thoughts about this policy, please write – DHA.

Dear Publisher:
Effective February 1, 1975 our prices will increase by approximately 10%. This revision is necessary because of the following:

A. Announced hike in newsprint cost, effective January 1, 1975, which is more than 13%.
B. A wage adjustment for our employees, which will be under the 12% rise in the Nation's cost of living index.
C. The escalating cost of supplies, some of which have gone up more than 100% during the past year.

We are trying to keep your printing costs at a minimum even though our rate of return continues to diminish. We are aware of the difficulties this will present to you, but have no alternative if we are to provide the quality and service necessary to produce your publication.

Ross P. Redmond
Redmond Press

Redmond Press is the company that actually prints each issue of Creative Computing. *Despite increased printing costs, it is our intention to hold the line on subscription prices. We also expect to gradually increase the size of each issue – dig this one gang compared to the last two – and the next one will be even bigger!! Just find someone else who's doing that for you in the middle of a recession – DHA.*

BACK ISSUES – POSTERS

Both back issues are completely sold out. Please do not write or send money for copies. Posters are all gone too.

CREATIVE COMPUTING Editorial

Birth of a Magazine

Early in my days as Education Marketing Manager at Digital Equipment Corporation, it became apparent that DEC was not communicating very well with its educational users and communication among users was virtually nonexistent. Consequently, I started *EDU* to act as a communication vehicle between DEC, its users, and prospective users. The first issue of *EDU* appeared in the Spring of 1971.

Over the years *EDU* flourished and grew into a 48-plus page magazine. However, there were certain aspects of educational computing which *EDU* could not satisfactorily address. In particular, school users, both college and elementary/secondary, need far more classroom activities, exercises, problems, and ideas than are available in textbooks and other magazines. Also, there ought to be a discussion of the social aspect of the computer, its effect on jobs, medical care, privacy, and the like. Furthermore, what about the user of non-DEC computers? Clearly to be responsive to these needs another vehicle was needed. Thus, *Creative Computing* was born, at least as an idea.

Since *Creative Computing* was intended to be responsive to the entire educational computing-using community and also schools who wished to expose their students to computer technology but could not afford the hardware, I reasoned that the National Science Foundation ought to be willing to provide some funding. My ideas were met with a great deal of encouragement, however, when it came to actually allocating some dollars I got passed from one office to another and eventually out the door.

NIE and OE were also very optimistic at first, but came up with no funds, even with some minor prodding from Senator Edward Brooke who was very helpful to us. Since I had prepared a rather extensive (and expensive!) proposal for the NSF, I decided to rework it slightly and approach various philanthropic foundations that were interested in education and technology. I ultimately submitted various versions of the proposal to 36 foundations. Twenty-eight sent a standard form letter refusal usually with a paragraph encouraging us to continue even though they couldn't help us financially. Two even sent the proposal back in case we wanted to use it elsewhere (that was, in fact, a very nice gesture since each proposal cost about $3.50 to print). Four foundations sent letters which indicated that someone had actually read more than just the abstract, however, still no money. And four foundations didn't reply at all (including two big, well-known ones that put out lots of glossy PR about the wonderful job they're doing).

Concurrent with the funding requests to foundations, I was also contacting various educational leaders to secure their endorsement of *Creative Computing* for other funding requests in the future. Out of 36 educators contacted, 33 endorsed the objectives of *Creative Computing* and allowed us the use of their signature on funding proposals. Armed with this additional high-powered ammunition, I printed more proposals (Variety 3) aimed at industry (makers of computers, peripherals, textbooks, etc.). One hundred and six companies were contacted. I'm afraid that activity lowered my rather high opinion of U. S. industry by several giant steps. Of the 106 companies, only seven bothered to reply at all! With the exception of three companies, the responses were negative.

I felt that three companies were not a broad enough base to put together a sponsorship program for *Creative Computing*, hence I decided to gut it alone on a shoestring out of my own pocket. (The three companies did render aid in various ways — advertising, mailing flyers, etc., and they deserve recognition: Educomp Corp., Hartford, CT; Hewlett-Packard, Cupertino, CA; and Computra, Munice, IN). However, most of the money and effort to print, address, and mail the initial 11,000 flyers came from me along with a handful of hired and voluntary high school helpers.

Response to the flyers was excellent; about 850 people subscribed before we even published an issue. However, about two-thirds of them asked to be billed (or invoiced). Imagine doing that by hand — which I did along with Andree Stone in Concord, MA. Ugh!

Between July and September 1974 was a busy time to say the least. Some of the concurrent activities going on included: writing 150 companies seeking advertising (2 responded), distributing flyers at 6 conferences (a waste of time — people take them and toss them in one continuous motion), preparing a subscription poster, preparing and mailing four different press releases to 224 other magazines (printed eventually by 19), purchasing mailing lists, contacting writers, artists, reviewers, and contributors, editing the first issue, finding and getting price quotes from typesetters, getting illustrations for articles, finding and getting quotes from printers, laying out the magazine, and answering on the average of 30 letters a week. All this while I was selling a house in Concord, Massachusetts, arranging for a move, purchasing a house in Morristown, N. J., moving, getting settled, finding schools for my kids, and learning the ropes of my new position as Marketing Manager-Education with AT&T. Whew! Disaster and/or divorce loomed nearer with every minute of every day.

Out of the scores of well-meaning volunteers (most of whom never seemed to have time when it was actually needed), three people emerged to do a bang-up editorial job on their respective sections. A. Kent Morton at Dartmouth edited the higher education section, Lynn Yarbrough coordinated and edited the reviews, and Walt Koetke of Lexington High School prepared the problems section. These people continue to be the nucleus of the editorial staff. Of course, many writers, artists, and other people contribute to the success of each issue.

October 7, 1974, the first issue rolled off the presses. Forty-three cartons, 52 pounds each — 8,000 copies. One basement plus family room full. Junior high school kids in the house around the clock. Label on each magazine, rubber stamped "ATTN: so and so", "Dear Comp. Center Director" or Librarian or — letter inserted, keep them in zip code order (ha!), sort again, tie into bundles, cart to the post office (oh for a large truck!), weigh, and mail.

Did it do the job? So far the world seems to like *Creative Computing*. Subscriptions keep coming in every day. Letters pile up and periodically get answered. The catastrophic disaster period seems to be ebbing away (although my wife hasn't noticed), and the magazine seems to have an established base of readers and contributors. Financially, an end to the descent into the red ink seems to be in sight and, if subscribers continue to join, the black ink can't be too far away.

Obviously, this is an unfinished story. Its ultimate conclusion will depend on people — contributors and readers. Perhaps I'll write the last chapter in three years or thirty years or never. Watch these pages!

David H. Ahl

> **"The reasonable man adapts himself to the world; the unreasonable man persists in trying to adapt the world to himself. Therefore, all progress depends upon the unreasonable man."**
>
> **G. B. Shaw**

psst! want a hot tip?

Do you have something to contribute to *Creative Computing*? Why not send it to us?

As you can tell from the first several issues we are focusing on activities and games for using computers in education and recreation. We also run articles, fiction, and humor on the role and effect of technology and computers on society and people. Our language is non-technical.

Contributions should be 500 to 3000 words. Typed, double-spaced. Include illustrations if possible (sharp black on white paper — not Xerox or other copies. Photostats are ok). Also, include photos if available (5x7 or large B+W — no slides or Polaroids).

If you want an acknowledgement, send a self-addressed stamped envelope.

In the future, about 40 to 50 percent of each issue will be devoted to a specific topic. The remainder will be diverse material with a broad interest. The special emphasis of upcoming issues is as follows.

Cover Date	*Article Due Date*	*Special Emphasis*
1975 Issues		
May-Jun	February 1, 1975	Expanding universe, extraterrestrial life, space exploration, time warp, 2001, etc.
Jul-Aug		Skip this issue in 1975
Sep-Oct	June 1, 1975	Early education, games for young kids, art and stories, value priorities
Nov-Dec	August 1, 1975	Privacy, computer data banks, medical data, criminal records, SSN
1976 Issues		
Jan-Feb	October 1, 1975	Mathematical games of all kinds. Also information about and activities using hand-held calculators
Mar-Apr	December 1, 1975	Computer art, plotter, display, image enhancement, scanners
May-Jun	February 1, 1976	Language arts, poetry by and about computers, word games
Jul-Aug	April 1, 1976	Artificial intelligence, robots, ELIZA, learning programs and games
Sep-Oct	June 1, 1976	General issue
Nov-Dec	August 1, 1976	Games special. All games in this issue, both manual and computer

NOT ONE

In the Nov-Dec 74 issue of *Creative Computing*, we challenged readers to write a good version of the game NOT ONE. Of the many versions received, the one printed below appears to be one of the best. It is by Robert Puopolo, Belmont Hill School, Belmont, Mass.

Complete playing instructions are given in the computer program itself. If you convert this to your version of BASIC, be alert for multiple statements on one line (separated by a colon) and extended IF statements (for example, IF-THEN-PRINT or IF-THEN-PRINT-ELSE-PRINT).

```
5 REM ROBERT PUOPOLO BELMONT HILL 6/25/74 "NOTONE"
7 PRINT:PRINT TAB(15);"NOTONE":PRINT
10 DIM T(50),R(10),C(10),L(12)
13 RANDOMIZE
15 INPUT "WOULD YOU LIKE THE INSTRUCTIONS";A$
20 IF A$="YES" THEN 35
25 IF A$="NO" THEN 100
30 PRINT:PRINT "ANSWER YES OR NO!!":PRINT:GOTO 15
35 PRINT:PRINT "THE GAME OF NOTONE IS PLAYED WITH"
40 PRINT "TWO PLAYERS AND A PAIR OF DICE. THERE ARE"
45 PRINT "TEN ROUNDS IN THE GAME. ONE ROUND CONSISTING"
50 PRINT "OF ONE TURN FOR EACH PLAYER. PLAYERS"
55 PRINT "(YOURSELF AND THE COMPUTER) ADD THE SCORE"
60 PRINT "THEY ATTAIN ON EACH ROUND, AND THE PLAYER"
62 PRINT "WITH THE HIGHEST SCORE AFTER TEN ROUNDS IS THE WINNER":PRINT
67 PRINT "ON EACH TURN THE PLAYER MAY ROLL THE TWO"
69 PRINT "DICE FROM 1 TO N TIMES. IF T1 IS THE TOTAL OF DICE ON"
72 PRINT "THE ITH ROLL, THEN THE PLAYERS SCORE FOR THE TURN IS"
75 PRINT "T(1)+T(2)+T(3)+........+T(N). HOWEVER,"
77 PRINT "AND HERE'S THE CATCH, IF ANY T(I)IS EQUAL TO T(1) THEN "
80 PRINT "THE TURN IS OVER AND HIS SCORE FOR THAT ROUND IS ZERO"
82 PRINT "AFTER EACH ROLL THAT DOESN'T EQUAL T(1), THE PLAYER CAN "
88 PRINT "DECIDE WHETHER TO ROLL AGAIN OR STOP AND"
90 PRINT "SCORE THE NUMBER OF POINTS ALREADY OBTAINED."
100 FOR T=1 TO 10:PRINT:PRINT "ROUND ";T
110 X=X+1:R1=INT(6*RND(X))+1
115 R2=INT(6*RND(X))+1:PRINT R1+R2
120 IF X>1 THEN 130
125 T(1)=R1+R2:GOTO 135
130 T(X)=R1+R2:IF T(1)=T(X) THEN
      PRINT "YOU GET A ZERO FOR THIS ROUND":X,T1=0:GOTO 200
135 T1=T1+T(X)
140 INPUT "ROLL AGAIN ";B$
145 IF B$="YES" THEN 110
150 IF B$="NO" THEN R(T)=T1:X,T1=0:T(A)=0 FOR A=1 TO 50:GOTO 200
160 PRINT:PRINT "ANSWER YES OR NO!!":PRINT:GOTO 140
200 PRINT:PRINT TAB(15%);"COMPUTERS MOVE":PRINT
201 RESTORE
202 R1=INT(6*RND(X))+1:R2=INT(6*RND(X))+1
204 READ L(D) FOR D=2 TO R1+R2
205 FOR C=1 TO L(D):IF C=1 THEN 215
210 R1=INT(6*RND(X))+1:R2=INT(6*RND(X))+1
215 PRINT "COMPUTER'S ROLL"C":";R1+R2
220 IF C>1 THEN 230
225 T(1)=R1+R2:GOTO 242
230 T(C)=R1+R2
235 IF T(C)=T(1) THEN PRINT "THE COMPUTER GETS A ZERO FOR THE TURN"
      :T1=0
      :GOTO 245
242 T1=T1+T(C):NEXT C
245 C(T)=T1:X,T1=0
250 C2=C2+C(T):C1=C1+R(T)
253 PRINT:T(B)=0 FOR B=1 TO 50
255 IF T=10 THEN PRINT "FINAL SCORE":PRINT
260 IF C2>C1 THEN PRINT "COMPUTER: ";C2,"YOU: ";C1 ELSE
      PRINT "YOU: ";C1,"COMPUTER: ";C2
300 NEXT T
305 DATA 18,18,9,9,6,6,6,9,9,18,18
310 PRINT:PRINT "SCORING SUMMARY":PRINT
315 PRINT TAB(17);"YOU":PRINT
320 PRINT E,R(E) FOR E=1 TO 10:PRINT:PRINT "TOTAL: ";C1:PRINT
325 PRINT TAB(17);"COMPUTER":PRINT
330 PRINT E,C(E) FOR E=1 TO 10:PRINT:PRINT "TOTAL: ";C2:PRINT
```

```
               NOTONE

WOULD YOU LIKE THE INSTRUCTIONS? YES

THE GAME OF NOTONE IS PLAYED WITH
TWO PLAYERS AND A PAIR OF DICE. THERE ARE
TEN ROUNDS IN THE GAME. ONE ROUND CONSISTING
OF ONE TURN FOR EACH PLAYER. PLAYERS
(YOURSELF AND THE COMPUTER) ADD THE SCORE
THEY ATTAIN ON EACH ROUND, AND THE PLAYER
WITH THE HIGHEST SCORE AFTER TEN ROUNDS IS THE WINNER

ON EACH TURN THE PLAYER MAY ROLL THE TWO
DICE FROM 1 TO N TIMES. IF T1 IS THE TOTAL OF DICE ON
THE ITH ROLL, THEN THE PLAYERS SCORE FOR THE TURN IS
T(1)+T(2)+T(3)+........+T(N). HOWEVER,
AND HERE'S THE CATCH, IF ANY T(I)IS EQUAL TO T(1) THEN
THE TURN IS OVER AND HIS SCORE FOR THAT ROUND IS ZERO
AFTER EACH ROLL THAT DOESN'T EQUAL T(1), THE PLAYER CAN
DECIDE WHETHER TO ROLL AGAIN OR STOP AND
SCORE THE NUMBER OF POINTS ALREADY OBTAINED.

ROUND  1
 6
ROLL AGAIN ? YES
 4
ROLL AGAIN ? YES
 9
ROLL AGAIN ? YES
 7
ROLL AGAIN ? YES
 11
ROLL AGAIN ? NO

               COMPUTERS MOVE

COMPUTER'S ROLL 1 : 5
COMPUTER'S ROLL 2 : 7
COMPUTER'S ROLL 3 : 4
COMPUTER'S ROLL 4 : 6
COMPUTER'S ROLL 5 : 9
COMPUTER'S ROLL 6 : 5
THE COMPUTER GETS A ZERO FOR THE TURN

YOU:  37       COMPUTER:  0

ROUND  2
 3
ROLL AGAIN ? YES
 9
ROLL AGAIN ?

ANSWER YES OR NO!!

ROLL AGAIN ? YES
 9
ROLL AGAIN ? YES
 7
```

```
ROUND  10
 8
ROLL AGAIN ? YES
 6
ROLL AGAIN ? YES
 10
ROLL AGAIN ? YES
 5
ROLL AGAIN ? YES
 5
ROLL AGAIN ? NO

               COMPUTERS MOVE

COMPUTER'S ROLL 1 : 3
COMPUTER'S ROLL 2 : 4
COMPUTER'S ROLL 3 : 9
COMPUTER'S ROLL 4 : 10
COMPUTER'S ROLL 5 : 8
COMPUTER'S ROLL 6 : 8
COMPUTER'S ROLL 7 : 9
COMPUTER'S ROLL 8 : 7
COMPUTER'S ROLL 9 : 9
COMPUTER'S ROLL 10 : 8
COMPUTER'S ROLL 11 : 6
COMPUTER'S ROLL 12 : 3
THE COMPUTER GETS A ZERO FOR THE TURN

FINAL SCORE

YOU:  345      COMPUTER:  299
```

Computing Factorials -- Accurately

by Walter Koetke
Lexington High School, Mass.

Multiple precision arithmetic is a topic that can easily capture the imagination of almost anyone interested in computing. Today's programming languages and even hand held calculators normally provide enough precision to satisfy the requirements of most users, so this topic is really most appropriate for those intrigued by the challenge of creative computing. Perhaps because multiple precision arithmetic is not studied by all students, introductory literature relating to the topic is very sparse. If you encounter a good reference, be sure to note it as the topic is rarely given more than two or three cursory pages.

Calculating factorials is a standard example in elementary programming courses. Although a good example of the technique required to compute a product, the fact that only a few factorials can be calculated exactly before being subjected to round-off error is usually ignored. Actually, not too many factorials can be computed before the arithmetic limits of BASIC are reached. A typical program that correctly calculates the factorial of an entered value is:

```
10 INPUT N
20 LET F=1
30 FOR M=1 TO N
40 LET F=M*F
50 NEXT M
60 PRINT F
70 END
```

Given a non-negative integer N, then N! (N factorial) is defined as:

if N O, N! = N(N-1)(N-2) . . . 1
if N = O, N! = 1

If only 6 significant digits are available, the results of this program are subject to round-off error for all values of N greater than 11. If 10^{38} is the upper limit of the available numbers, then this program can not even approximate the factorial for any N greater than 33. Even if 10^{99} is the upper limit available, the factorial can not be approximated for any N greater than 69. However, using multiple precision arithmetic, we can extend these limits to whatever extreme we choose.

To compute factorials more accurately, we must develop an algorithm for multiple precision multiplication. The most straightforward algorithm is that which we use when multiplying with pencil and paper. Hand calculation has many stumbling blocks, but a limited number of digits or a limited range of values are not among them.

Consider computing the product 7 x 259. You begin by multiplying 7 x 9, and although the product is 63 you only write down the 3 and "carry" the 6. After next multiplying 7 x 5, you add this "carry" to the product and obtain 41 — and again you write down the 1 and "carry" the 4. And so forth . . . After each individual multiplication, you record the units digit and "carry" those that remain.

To write a BASIC program that does multiple precision arithmetic using this same algorithm, one need only be able to separate the units digit of a product from the "carry". If P represents the product of two positive integers, then:

carry = INT(P/10)
and
units digit of P = P - 10*(carry)

Let's now apply this algorithm to the larger problem of computing the factorial of any positive integer. To do this we will write a program similar to the very brief example already given. However, the product shall be represented by the subscripted variable F, *each subscripted value representing a single digit of the product.* One program that does this is:

```
10 DIM F(50)
20 LET L=50
30 INPUT N
40 FOR I=2 TO L
50 LET F(I)=0
60 NEXT I
70 LET F(1)=1
80 FOR M=1 TO N
90 LET C=0
100 FOR I=1 TO L-1
110 LET F(I)=F(I)*M+C
120 LET C=INT(F(I)/10)
130 LET F(I)=F(I)-10*C
140 NEXT I
150 NEXT M
160 FOR I=L TO 1 STEP -1
170 PRINT F(I);
180 NEXT I
190 END
```

Notice that:

1. The program will compute factorials that can be expressed using no more than 50 digits. This restriction can be decreased by using a larger value in the DIM at line 10 and making a corresponding change in the value of L at line 20.

2. The product of two integers and the addition of the previous carry is completed at line 110. The next carry is calculated in line 120 and the unit's digit of the product is obtained in line 130. If you understand these three lines, you understand the fundamental idea of multiple precision multiplication.

3. All 50 (or L) digits of the product are always printed. This isn't wrong, but leading zeroes look peculiar.

Two sample runs of the program appear as:

```
RUN

? 5
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1
 2  0

READY

RUN

? 40
 0  0  8  1  5  9  1  5  2  8  3  2  4  7  8  9  7  7  3  4  3  4  5  6
 1  1  2  6  9  5  9  6  1  1  5  8  9  4  2  7  2  0  0  0  0  0  0  0
 0  0

READY
```

Now stop reading and try running this program. Can you improve it? Increase the number of digits in the product. Print only those digits of the product that are significant. Print the product without spaces between each digit. Try to do these things *before* you continue reading — and if you can't use a terminal you can still write the required program changes.

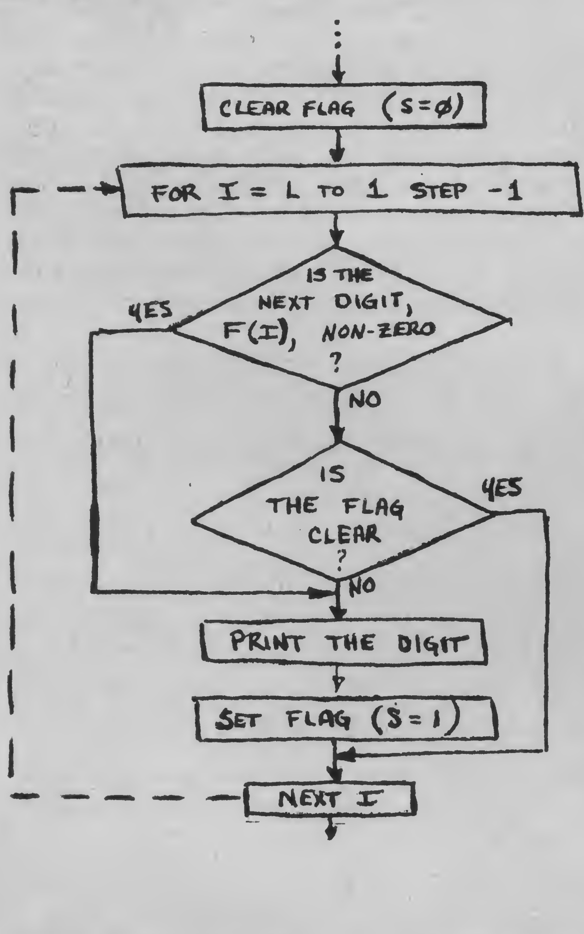
If you were successful in completing the suggested improvements, then read fast for awhile. Increasing the number of digits in the product from 50 to 150 can be done with:

```
10 DIM F(160)
20 LET L=160
```

The only limit to the number of digits is the upper limit of the subscripts available in the BASIC you are using.

Printing the product without spaces between digits can be done in several different ways — most of which are a function of the version of BASIC you are using. Since this has little to do with multiple precision arithmetic, removing the spaces remains your problem.

Deleting leading zeroes in the printed product doesn't have much to do with multiple precision arithmetic either, but let's delete them anyway. This is not being done arbitrarily, but because it provides a very good example of the use of a "flag" within a program. Quite simply, we will use one variable, say S, as a flag to indicate whether a non-zero digit has been printed. All zeroes can then be ignored rather than printed unless a non-zero digit has been printed. This is represented in flow chart form as:



This algorithm for omitting leading zeroes is added to the program by:

```
160 LET S=0
170 FOR I=L TO 1 STEP -1
180 IF F(I)>0 THEN 200
190 IF S=0 THEN 220
200 PRINT F(I);
210 LET S=1
220 NEXT I
230 END
```

Two sample runs of the modified program appear as:

```
? 5
 1  2  0

READY

RUN

? 100
 9  3  3  2  6  2  1  5  4  4  3  9  4  4  1  5  2  6  8  1  6  9  9  2
 3  8  8  5  6  2  6  6  7  0  0  4  9  0  7  1  5  9  6  8  2  6  4  3
 8  1  6  2  1  4  6  8  5  9  2  9  6  3  8  9  5  2  1  7  5  9  9  9
 9  3  2  2  9  9  1  5  6  0  8  9  4  1  4  6  3  9  7  6  1  5  6  5
 1  8  2  8  6  2  5  3  6  9  7  9  2  0  8  2  7  2  2  3  7  5  8  2
 5  1  1  8  5  2  1  0  9  1  6  8  6  4  0  0  0  0  0  0  0  0  0  0
 0  0  0  0  0  0  0  0  0  0  0  0  0  0

READY
```

If you tried running this program as suggested, you probably discovered something else that needs to be improved — the speed of computation. The present form of the program always multiplies each of the integers 1 through N by each of the variables F(1) through F(L-1). Thus when N = 100 and L = 160 as in the last sample run, the computation loop (lines 110 - 130) is repeated 15,900 (100*159) times. Even when N = 5 this loop is repeated 795 (5 x 159) times, and that's a lot of work to compute 1*2*3*4*5. This excessive computation can be eliminated by making use of *a pointer*, a very important idea in computing. Essentially, we will use another variable, say P, to point at the left most non-zero digit in the product. We can then multiply each of the integers 1 through N by each of the variables F(1) through F(P). When N = 5, this reduces the number of repetitions of the computation loop from 795 to 6, or more than 99%. When N = 100, the reduction is from 15900 to 6834, or about 57%. Clearly a pointer provides a worthwhile savings. Try to verify these reduced counts before you leave this topic.

If you are unfamiliar with the concept of a pointer, be sure you read these paragraphs very carefully. After you become familiar with this idea, teach your students about pointers if you're a teacher, or teach your teacher about pointers if you're a student. Who learns a significant idea first is not nearly so important as having everyone eventually understand the idea.

To implement the use of a pointer in our factorial program, begin with the initial value, P = 1. We then want to multiply each of the integers 1 through N by each of the variables F(1) through F(P). Thus line 100 should be changed to read FOR I = 1 TO P. The pointer does not alter the multiplication algorithm in lines 110 through 130, but after the NEXT I in line 140, we must examine the carry to see if the pointer is to be incremented. If the carry is non-zero, we increment the pointer, perform the carry, and then repeat the procedure. If the carry is zero, we can continue with NEXT M.

The instructions needed to do this are:

```
142 IF C=0 THEN 150
143 LET P=P+1
144 LET F(P)=C
145 LET C=INT(F(P)/10)
146 LET F(P)=F(P)-10*C
147 GOTO 142
```

Finally, since the product contains exactly P digits, then the print loop beginning at line 170 can be changed to read FOR I = P TO 1 STEP -1.

Our multiple precision factorial program, complete with all modifications discussed, now appears as:

```
10 DIM F(160)
20 LET L=160
30 INPUT N
40 FOR I=2 TO L
50 LET F(I)=0
60 NEXT I
65 LET P=1
70 LET F(1)=1
80 FOR M=1 TO N
90 LET C=0
100 FOR I=1 TO P
110 LET F(I)=F(I)*M+C
120 LET C=INT(F(I)/10)
130 LET F(I)=F(I)-10*C
140 NEXT I
142 IF C=0 THEN 150
143 LET P=P+1
144 LET F(P)=C
145 LET C=INT(F(P)/10)
146 LET F(P)=F(P)-10*C
147 GOTO 142
150 NEXT M
160 LET S=0
170 FOR I=P TO 1 STEP -1
180 IF F(I)>0 THEN 200
190 IF S=0 THEN 220
200 PRINT F(I);
210 LET S=1
220 NEXT I
230 END
```

Any additional improvements are left to you. Although several are possible, more efficient use of the variables in the F array is likely to be the most dramatic. The example program uses one variable to represent one digit. By allowing each variable to represent two or three digits, program speed is essentially doubled or tripled. But then there are additional problems when the product is printed, and possibly additional problems within the computation loop. Does the example have any theoretical limit on the factorials that can be computed? Does it have any realistic limits?

The topic of multiple precision arithmetic will be further explored in a subsequent column. Now that you understand multiple precision multiplication, only the operations of addition, subtraction and division separate you from successfully computing 2 and to at least half million digit accuracy.

(To further test your knowledge of multiple precision arithmetic, why not try Contest Problem 3. — Ed.)

Glum Glossary

Punched card: A short piece of 80-channel paper tape.

Program: The footprints of hundreds of bugs. Once the bugs are eliminated, the program is all that's left.

Puzzles and Problems for Fun

▶ Mrs. Canton wanted to buy all the grocer's apples for a church picnic. When she asked how many apples the store had, the grocer replied, "If you added 1/4, 1/5 and 1/6 of them, that would make 37." How many apples were in the store?

▶ Donna bought one pound of jellybeans and two pounds of chocolates for $2. A week later, she bought four pounds of caramels and one pound of jellybeans, paying $3. The next week, she bought three pounds of licorice, one pound of jellybeans and one pound of caramels for $1.50. How much would she have to pay on her next trip to the candy store, if she bought one pound of each of the four candies?

▶ Take a 3-digit number like 200, reverse it (002) and then multiply the two numbers. The result, 400, is a perfect square (20* 20 = 400). Find all such 3-digit numbers.

Bill Morrison
Sudbury, Mass.

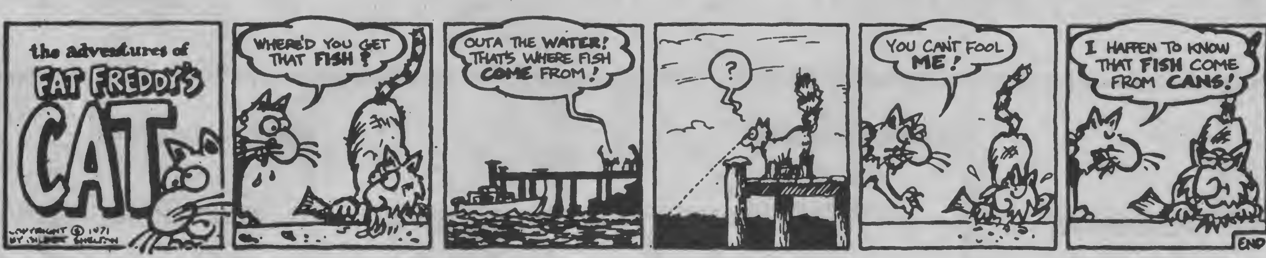
▶ Can you put nine pigs in four pens so that there are an odd number of pigs in each of the four pens?

* * *

✱ If you have a favorite puzzle, perhaps we can print it here. Send it along! ✱

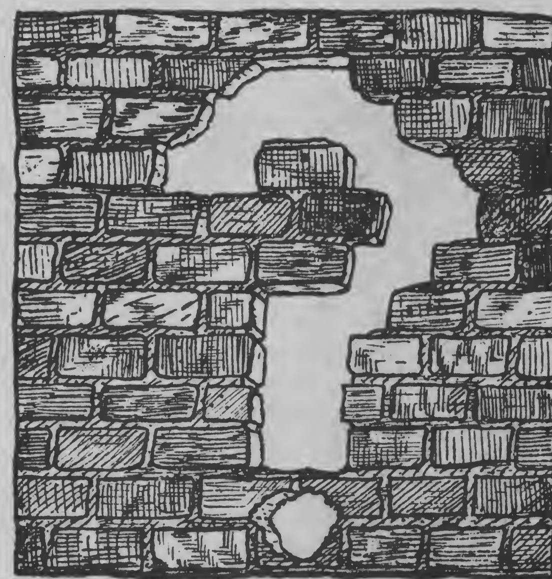

theme song from the
FABULOUS FURRY FREAK BROTHERS
COPYRIGHT © 1971 BY GILBERT SHELTON
WE WENT LOOKIN' FOR TH' GOOD LIFE, BUT WE SHOULDA STOOD IN BED,
'CAUSE THE STUFF WE ATE & DRANK & BREATHED IT PLUMB CLOGGED UP OUR HEAD,
AND THE THINGS WE SAW ON TEEVEE, ON THE NEWS EACH NIGHT AT TEN,
WERE ENOUGH TO MAKE YOU WANT TO GRAB YOUR TOWEL AND THROW IT IN!
WELL, SOMETIMES OUT IN THE COUNTRY YOU CAN STILL SEE BIRDIES SING,
YOU CAN'T HEAR 'EM, THOUGH, 'CAUSE JET AIRLINERS DROWN OUT EVERYTHING,
AND SAY, WHATEVER HAPPENED TO THOSE SIMPLE THINGS WE PRIZED?
THERE'LL BE A LOTTA CHANGES MADE NEXT MONTH WHEN WE GET ORGANIZED!!


WE WENT LOOKING FOR AMERICA
BUT WE COULDN'T FIND A THING,
'CAUSE THE AIR WAS THICK & YELLOW
AND OUR EYES BEGAN TO STING,
CONK
SPLAT
THUMP
AND THE TRAFFIC WAS SO HEAVY
THAT YOU COULDN'T TURN YOUR HEAD,
BLAST!
HONK HONK
BEEP BEEP
IF YOU SLOWED DOWN BELOW EIGHTY
YOU'D BE DEAD!
WELL, WAY OUT THERE IN THE COUNTRY
YOU CAN STILL HEAR BIRDIES FLY,
YOU CAN'T SEE 'EM, THOUGH, 'CAUSE NEON
SIGNS AND BILLBOARDS FILL THE SKY,
VOTE FOR SLEEZE
GUARANTEED USED CARS
MOTEL
LIQUOR
BEER
DON'T WALK
GIANT CHILI DOGS
TV REP
ONE WAY
LOANS
WINE
JEWELRY
BUT WE DON'T LIE AWAKE AT NIGHT,
AND WE DON'T EVEN SWEAT,
WE JUST LIGHT OURSELVES ANOTHER
CIGARETTE.
the Saviour
THE CHAIRMAN
BOTH THE SAVIOUR AND THE CHAIRMAN
SAID WE ALL LIVE IN THE SEA;
STICK YOUR NOSE UP IN THE AIR, MAN,
AND THE TRUTH WILL COME TO THEE:
WE ARE FISH UPON THE SURFACE; SEE
THE SCUM GO SAILING BY!
WON'T SOMEONE TURN ON THAT GREAT
BIG FISHBOWL FILTER IN THE SKY!?
WELL, I DON'T CARE
IF I DO DIE
(DOO DAH)
DO DIE
(DOO DAH)
DO DIE,
GONNA STICK MY NOODLE
IN THE OLD GRAY AIR
TILL I DO DIE,
HEY!
SMOKANE
PENGENE
LOS ARTERIOSCLEROSIS
SMEJENA
CARCINOMA CITY
PEENIX
PEETUMA
SHIT LAKE CITY
SANTA FETID
SMOKELAHOMA CITY
AWE STINK
DETOILET
NO AIR LEANS
HEAVELAND
TRASHVILLE
IMMOBILE
SHITTSBURGH
BURNINGHAM
STENCHMOND
MIASMA
NO REFILL ADELPHIA
SWILLMINGTON
NEW YECCH CITY
FARTFORD


the adventures of
FAT FREDDY'S CAT
COPYRIGHT © 1971 BY GILBERT SHELTON
WHERE'D YOU GET THAT FISH?
OUTA THE WATER! THAT'S WHERE FISH COME FROM!
?
YOU CAN'T FOOL ME!
I HAPPEN TO KNOW THAT FISH COME FROM CANS!
END

YOU ARE A BUSINESSMAN...

A COMPUTER expert/economist has come to you with a mathematical model of a means of so stimulating the economies of urban ghettos that they would cease to be a problem to the nation or themselves in a decade. The most conserfative estimates are that the money saved in local and federal governmental costs as well as by ghetto dwellers would equal the savings of 25% of the gasoline used in the country.



A very large sum of money is needed to perfect the economic and mathematical relationships to enable the computer to be used to direct the program, as well as to act as seed money for the enterprises, provide funds for education and training, inventory, refurbishing of the areas, and a sales and marketing effort.

It's more money than you and your associates have; it is essential that government interference be avoided; you cannot supply the enormous collateral the banks would demand for a loan. You're going to have to sell stock in it, millions of shares of stock at a low price to raise the millions of dollars needed. The idea is a necessary one, and will ultimately save taxpayers millions, maybe billions of dollars each year, as well as providing ghetto dwellers the equality of opportunity we worship as one of our value-beliefs.

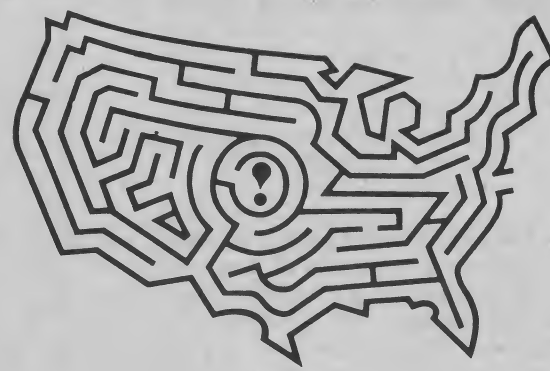
PREPARE a statement inviting people to invest in buying stock -- a few shares or a lot of shares. Point out the need, the potential in the action, the possibility that the ultimate return will be in tax savings and satisfaction of seeing the disadvantaged help themselves, the dangers and bad experience of leaving such a task to government -- everything you can to induce someone to see the gain to himself, if not in dollars in something else, of such an investment.

WHEN you have made up your sales pitch for the prospective corporation, try it out on your parents and some of their friends. Ask them if they would invest in such a company, and why.

WHAT WERE THE RESULTS OF YOUR SURVEY?

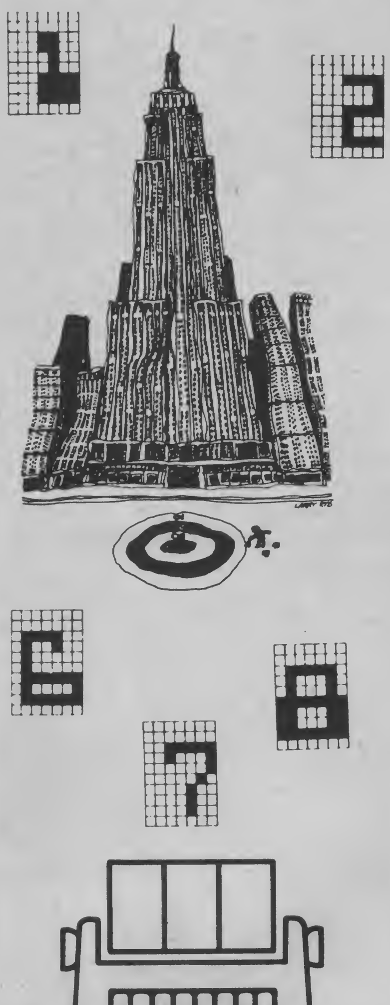
WHAT DO THESE RESULTS INDICATE AS FAR AS A MESHING OF ADVANCES IN TECHNOLOGY AND OUR INSTITUTIONS?

THERE'S MORE THAN MEETS THE EYE



Don't get the idea that the question is one of profits versus idealism. The question boils down to using technological advances -- of being able to apply scientific knowledge to solving our social problems -- some of which were caused by technological advances -- with the same efficiency we can supply cars, or frozen foods, or polyester clothing.

Here are six questions that Harvard University is studying about this problem. Just on the basis of what you know and feel -- and what you've found out doing some of the exercises in this book -- what do you think might be a way of answering these questions and solving the problems they describe?



1. How do the organization and goals of corporations -- directed at profits to provide money to plow back into the business, salaries for top-notch people, and a profit for the investors -- affect the ability of our society to provide goods and services that are socially desirable, but not necessarily profitable to any one group in society?

2. What does it cost our society to concentrate so much effort on economic growth to provide jobs for as many people as possible rather than investing some of our capital in social development?

3. How do you reconcile the incentive to individuals of profits and achievement with the public need for goods and services that may not return either to those providing it?

4. How can you reconcile the all too frequent fight between what is desirable to us as individuals for our own welfare and what is desirable to us as members of society from the standpoint of welfare?

5. Are the roles of government and private enterprise due for some major changes because of the short-term as well as the long-term effects of technology?

6. How can advertising and other forms of communication be used in public education to make people aware of the need to make decisions affecting these questions before the pressures of value changes, social changes, and economic and political stresses tear our society apart?

Career Education. Will it Last?

By Joyce L. Kennedy

Career education - the name of the "turnaround" movement needed in our schools to prepare students for work as a major part of preparing them for living -- is the "in" education concept now. The question: Will it outlast previous educational concepts [remember the right to read" a couple of years ago]?

As an eminent educator, Harold Howe, a former Federal education commisioner and now education vp for Ford Foundation, told an audience, some fear that the career education concept "is so general that it runs the danger of being watered down into a mass of lip service activity."

For example, Dr. Howe noted that the chief state school officers have unanimously endorsed the idea but he "really wonders" if they "intend to go out there and do a job.") In rough translation - will school decision-makers put their money and muscle where their mouths are?)

DR. HOWE'S RESERVATIONS recognize the moon-launch (size job ahead if career education is to flourish. Just a few of the massive tasks: making drastic changes in teacher training institutions to reflect a career education emphasis; retraining the current administrative and teaching corps; rewriting classroom instructional materials; establishing home-school-community workplace tie-ups; convincing employers to co-operate by hiring students and teachers for brief periods; getting taxpayers to foot the bill; and persuading administrators of school budgets to spend the money on career education programs. Society, too, must stop thinking that the plumber who learned his skills in an apprenticeship program does not deserve the dignity equal to that given the physicist who learned his skills in college.

In spite of the fearsome number of built-in problems, there are convincing reasons for believing that career education is not a trendy fad, but a powerful idea destined to root and multiply.

For one thing, the needs of people for assistance in career preparation and development are becoming unmanageable in our present setup. Without substantial educational restructuring both "overchoice" (the task of selecting from 28,000 known occupations) and "underchoice "the denial of opportunity through lack of knowledge of and preparation for work) will get worse as we progress further into the postindustrial age. The individual without skills will become a dreg on the market, even more so than today. If increasing numbers of people continue to be highly educated for living, but are without skills to pay for living, the social unrest now will seem harmonic in retrospect. Education is the logical institution to lead the way in providing job skills, positive work attitudes and "amplechoice."

ANOTHER POINT focuses on fairness. The needs of the real majority of citizens must be served as well as those of the academically elite minority. Between 75-80 percent of people are not likely to find jobs requiring a 4-year college degree in years ahead, according to the Federal Labor Department.

THE WORK ETHIC can be restored. People can find satisfaction in achievement. The manpower needs of our nation can be met while maintaining individual freedom of choice. Dignity can be given to all occupations. Career education holds promise of doing these things. But, as noted in a Federal education handbook:

"Career education is an idea whose time has come. Still, it will not come to your community until someone grasps the initiative and brings it there. If someone, why not you? If sometime, why not now?"

Reprinted with permission of *The Computer Educator*, Oct. 1974.

DP Salary Survey

Top managers of edp installations now earn an average of $537 a week, or $27,924 annually, according to the *1974 Weber Salary Survey on Data Processing Positions in the U.S.*, published by A.S. Hansen, Inc. of Lake Bluff, Ill. The 1974 average for managers rose 10.2 percent over 1973, the survey shows. Other job categories that showed impressive gains in 1974, with the 1973 increases in parentheses, were: systems programmers, 10.8% (4.8%); systems analysts, 8.7% (4.4%); keypunch operators, 6.5% (2.8%); and computer operators, 7.2% (5.6%). For applications programmers, the 1974 increase was 4.8%, compared with 5.1% in 1973.

Key to Your Future?

By Eleanor Corr
Keystone Junior College
La Plume, Pennsylvania

What turns YOU on? Law? Public Relations? Engineering? Writing? You don't want JUST a job, right? You want to look to a full and rewarding life! This means that you will be working towards that goal which you expect will assist you in achieving self-realization and self-fulfillment.

Career education is regarded as a means of fulfilling that goal.

However, to make appropriate choices concerning educational goals, occupational or vocational careers, you must first acquire knowledge about your aptitudes, interests, and abilities. Consequently, individual growth and development is predicated on freedom of choice, needed skills and knowledge.

If you have bothered to pick up and read through this magazine at all, you have exhibited some interest in the computer field and whether your interests are in the field of business or science, your options are many.

So what about computer-related careers, you ask?

The field of computers is new and dynamic and extends to all parts of society. The computer is not constrained in its use. We're all aware of its effect on government, national defense, medicine, industry, business, and education. YOU can be a part of this new field and still be true to your own "inner calling."

Let's look at the following: RESEARCH (basic research working on applications from a humanitarian point of view such as medicine, weather, energy, space), ENGINEERING (applied research, the design and development of equipment), MARKETING RESEARCH (determining the needs of industry, education, etc.), PROGRAMMERS (systems analysis and design, development of compilers, designing applications), TECHNICAL WRITERS (writing the documentation for product applications), SYSTEMS ENGINEERING (assisting customers in use of equipment, installation of systems), FIELD ENGINEERING (maintenance of systems), MARKETING REPRESENTATIVES (selling computer-related products, conducting presentations), ADMINISTRATIVE positions (maintaining customer relations, planning, organizing work, schedules, etc.), LAW (data processing specialist in corporate law), EDUCATION (teaching data processing, computer science, telecommunications).

Only a few career fields have been mentioned, but I'm sure you can add to that list. Of course, each of those careers which I have listed contain other dimensions. For instance, PROGRAMMERS can be systems analysts or systems designers, or perhaps one programmer may do the "coding" from the systems design of another programmer.

And the "piece de resistance?" Because of the rapid pace of technological change in this field, individuals must constantly be engaged in continuing their professional education. You can be at the forefront of it all where it's important to learn and to know! Today, a career involving computer applications is not only financially rewarding, but challenging and fulfilling as well. Is one in your future?

COMPUTERS IN AUTO MAINTENANCE
Wayne Block runs down the computer maintenance checklist before servicing this police car in the Sparks, Nev., city garage. The computer system automatically schedules maintenance work for each of Sparks' 200-plus vehicles, as well as handling the city budget, payroll and even the quarterly sewer assessment. (Photo IBM)

A "Young" Business

Creative Computing just heard last week about three boys in Portland, Oregon who have organized a computer software company and are providing part-time services to local firms. All three boys are under 18 and have one or two years more of high school.

Apparently they had some minor legal problems getting started since they weren't allowed to operate out of a private residence and had to rent an office. Hence the problem: they were all too young to sign contracts so two parents had to co-sign. But they don't seem to be having any trouble with the work or getting it either, indeed, most of it is through referrals. All of their work is done in BASIC and if you want more information, you can write them directly:

Frank J. Barberis
Computa-Link Company
Oregon Pioneer Bldg., Room 302
Portland, Oregon 97204

A Computer Career For You?

by Judy Edwards

This article was written while Judy was at the Northwest Regional Educational Laboratory in Portland, Oregon. She is now a Research Assistant at the Lindquist Center, University of Iowa.

INTRODUCTION

The world of computers is rapidly changing. New hardware and software is being developed almost daily. The people who work in the computer center of the future will find the pace increasing. New ideas today will be obsolete tomorrow and they will need to learn rapidly about new developments.

The demand for skilled people will grow rapidly as technological projects grow. The competition for jobs may increase as the need for up-to-date technological knowledge is required.

For those who are planning a career in the computer world, careful planning for that career can insure them a secure place in an ever-changing environment. There will always be new things to learn and advancements for those who assess their abilities, build a good training and education program for themselves and finally survey the jobs thoroughly to find the right position.

Let's look at some of the factors involved in planning a career in computers and see how you can become more aware of what you would want to do in the field of data processing.

THE NEW TECHNOLOGICAL AGE

There will be a growing need for computers as information increases. Computers will have an ever increasing job to do in business, science, government, education and in the professinal world. In fact it is likely that many of us will use a simple form of computer in our homes within just a few years.

With the greatly increased appearance of computers, it seems accurate to predict that more and more jobs will exist. Many of the jobs and working environments will not be what they are today, however. Obviously, as the computer takes on more and more routine work, our work will be new and more interesting. We will learn new skills and find a challenge in keeping up-to-date on discoveries in computer technology whether we work directly with computers or not.

New Career Atmospheres

People who work directly with computers will have an inside picture of technology at work. They will see less and less paper work and more machine-stored information. They will learn about telecommunication systems — those which send computer-stored information over communication lines and television screens to millions of people. Computer personnel will work with and understand vast networks of computer systems which will serve a world-wide population. Specialists in computer science will see knowledge of all kinds computerized for rapid access. Library reference information, for example, will be computerized and referenced by pushing a button on a small table-size or pocket-size computer.

Whatever the application, computer center personnel will continually be learning and moving ahead in a more and more creative atmosphere. The new technologists will not only be concerned with the development of computing machines, but they will be concerned with better understanding of human processes so that man will be capable of relating those complex processes to the computer.

COMPUTERS IN HOSPITALS
CRT terminals are used in many hospitals to record medical information on patients being admitted. (Photo DEC)

The computer, in return, will objectively analyze and report what it sees. In this age learning may be a two-way process — from man to machine and from machine to man.

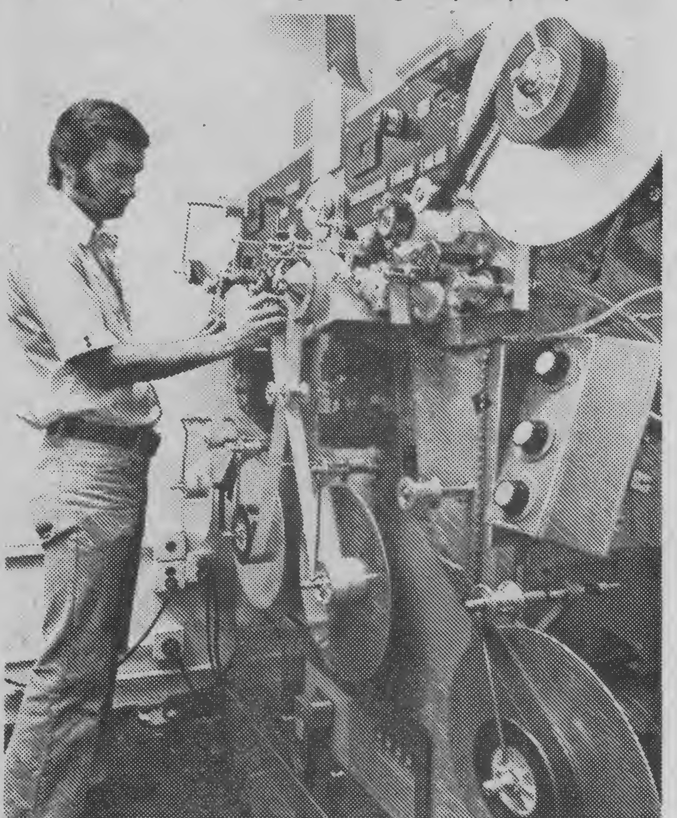
The Demand for People

For all its amazing ability, the computer has no power except that given it by people. People design and build computers, analyze and solve problems for the computer, and operate and service computing machines. People are essential to the new technology — people who are properly trained and experienced are in great demand.

Computer technology has grown so rapidly that there is a great shortage of technically trained people in all data processing jobs. The data processing industry is growing four to five times faster than any other American industry and in the process has created many unfilled jobs.

The demand for trained personnel will continue to be greater than the supply for several years to come. The number of computer installations in the United States alone promises to hit the 100,000 mark by 1975. By then, some two million people may be needed to fill jobs directly related to computers.

The largest category of computer-related jobs will be in the scientific fields: mathematics, physics, chemistry, and engineering. Equally important jobs will be open in accounting, business practices, and in the professions. Naturally, these are the people closest to the designing and scientific use of computers.



COMPUTERS IN THE MOVIE INDUSTRY
A technician at Consolidated Film Industries operates a printing machine, which makes projection copies of original negatives for theater and television use. An IBM System/7 monitors the printers to ensure that the original colors are faithfully reproduced in the prints. This process earned Consolidated Film Industries and IBM an Academy Award for technical achievement. (Photo IBM)

Technicians of all kinds will also find jobs. Operators of auxiliary equipment, key punch operators, clerks, and tape librarians are only a few. In computer manufacturing, there is a growing need for skilled technicians to aid designers and engineers in their work. In addition, the customer engineers who service and maintain computer hardware have become an essential group to all computer manufacturers.

Systems analysts, programmers and computer operators will be in great demand. In addition, jobs in sales and software development for computer manufacturers promise to increase quite rapidly.

IS A COMPUTER CAREER FOR YOU?

Before you can even begin to plan for a future career in data processing, you should ask yourself whether a computer career is the best career for you. Let's explore how you can begin to see whether you fit somewhere in the world of computers.

Knowing Your Abilities

Each person in a computer center should possess certain personal characteristics. Certain jobs, those of the systems analyst and computer center manager require the ability to communicate with other people. Some other jobs, such as data preparation and computer operation, require some mechanical skills. Programmers have less to do with people and more to do with reasoning and problem-solving. People in a computer center have one quality in common — the ability to solve problems logically and efficiently.

In looking into career possibilities, you should be aware of the personal abilities and interests you have which fit a particular job. Various positions in a computer center were briefly mentioned, but it would also be helpful to look into jobs outside the average computer center. Perhaps your interests are more compatible with another type of job: computer designer, electronics engineer, computer salesman, or customer engineer, for example.

One important step which is helpful to anyone just beginning to plan a career is to take stock of his own abilities and interests. Are you mechanically inclined? Do you enjoy working with people? More than you enjoy sitting at a desk and logically organizing the solution to problems?

This first step in looking at career choices is mostly up to you. You must first decide which of your interests or skills are most important. You can discuss the answers to these questions with your school job counselor who may have access to information about the abilities required for particular job. In addition, most schools also offer a variety of aptitude tests for clerical and operations skills and others for computer programming. You might want to take these tests as part of your plan to realize your best skills.

Deciding What You Want To Do

If you are interested in a career related to computers, you can begin now to decide what you like to do, what your skills might be, and how they fit the various jobs open to you. It is quite possible, as you learn more and gain more experience, that your skills or ambitions may change. In the world of computers, being ready for change is an advantage rather than a fault. But for now, you should gain some general knowledge of what you would like to do.

PLANNING YOUR EDUCATION AND TRAINING

Often the second question asked by anyone beginning to think about a career is what education or training must I have to get the job I want? Assume you have, by now, at least decided upon the general category of computer-related jobs you would like to try. That job or any jobs in the computer field requires a certain degree of general educational background.

Required education for a particular job may lead to a high school diploma, junior college degree (associate degree), college degree, or an advanced degree such as the masters degree or Ph.D. You can see there are several levels of education available to you. Each of the jobs in data processing will usually require some general education on one of these levels.

Getting training for a job can be quite different than getting an education for a job. Training implies more specific learning. In data processing, training usually means taking courses arranged by a vocational school, junior college, an employer or a computer manufacturer. Courses of this kind teach specific skills: computer operation skills, programming skills, or data preparation techniques, for example. Each job in the computer field also requires some specific training.

Let's look at the various degrees of education and training you may want or need to become proficient enough to begin working in a computer-related field.

Courses in High School

Computer science courses in high school at the present time prepare you for future training. Most high schools offer courses in computer concepts, elementary computer programming and some data preparation courses, such as beginning principles of key punching. These courses generally do not prepare you for a career in data processing. They do, however, allow you to explore the fundamentals of computer science so that you can more clearly decide which job you would most like to do.

Your high school years are ideal for exploring the specific jobs, the career possibilities and learning about the larger picture in the working world. Any computer science courses, math courses, English, and writing courses you take will prepare you for bigger decisions later. Thus, you can use high school as a stepping-stone to help answer some larger questions, such as what specific training do I need, which training school or college shall I choose, or what company shall I work for?

Type of Training	Length of Course	Courses Offered	Cost
Home-Study Schools	18 months	Computer Electronics	\$200-\$500
Commercial Data Processing Schools	60-80 hours 100-200 hours 400 hours 400-1,000 hours	Key Punching Computer Operation Computer Electronics Computer Programming	\$100-\$150 \$250-\$500 \$550-\$1,000 \$400-\$1,500
Junior College Data Processing Course	6 months-2 years	Key Punching Computer Operation Computer Electronics Computer Programming Systems Analysis	\$100-\$400
Computer Manufacturer Courses (employees of computer manufacturer)	2 days-6 weeks 6 months-2 years	Data Preparation Computer Operation Computer Programming Systems Programming Systems Analysis Management Principles Sales Software Design Customer Engineer	Usually free to employee of user or manufacturer

Vocational Training

Vocational courses (job-oriented courses) are becoming more and more popular. Educators and employers have begun to realize that a college degree is unnecessary for certain jobs. Many careers in data processing, in particular, require more specialized training and less of the general education acquired at colleges and universities.

It is not at all unusual to find many people without college degrees in computer-related jobs. Where did they get the necessary training to become data preparation specialists, computer operators, programmers or even systems analysts? At vocational training centers, at private business schools, in a two year job training program at a junior college, with a computer manufacturer or in an on-the-job training session paid for by a new employer.

A four or five year college or university program is not essential to begin a career in data processing. There are, of course, computer careers which require a college education and some highly specialized training. Those we will look at in the next section, "Going on to College."

Right now, let's look at some of the jobs which might be best suited to short-term training — jobs which do not require a college education. Some of the most likely are data preparation clerk, computer operator and computer programmer. The jobs of systems analyst can also be obtained without a degree but often requires some advanced study in business administration. Other jobs with computer manufacturers or related businesses which require only specialized training courses are customer engineer, equipment salesman and computer service representative. The schools which offer training for these jobs offer a variety of courses, but their costs vary according to the length and completeness of the course. The chart (previous page) will show you some types of vocational training schools, the courses they offer, and the general cost of the training.

If you think you would like to look into computer training after high school, you can check into enrolling in one of these schools. Notice that home study schools and commercial data processing schools are rather high in cost. The junior college courses, on the other hand, will cost only the price of tuition and books. Home-study courses are primarily meant for those who live in remote areas and have no other source of training.

A word of warning about commercial data processing schools — some of the schools are not as honest as they appear to be. If you decide on a data processing school for your training, check carefully into the school's reputation and see that they actually can train you for the job or that their tuitions are reasonable for what they offer. Again, you can consult your school job counselor, the computer science teacher in your school, or your school library for information on vocational training for computer careers.

Going on to College

A college degree is not essential for many jobs and careers in data processing, as you have seen. For some computer careers, however, a college education is either very helpful or even necessary to achieve a certain level of employment. Such specialized positions as systems programmer, software specialist, computer hardware designer and scientific systems analyst usually require a college degree.

If you have decided on a career in a specialized computer field — scientific data processing, computer design and manufacturing, or teaching computer science, for example — you would do well to explore now the many possibilities for a specialized education.

Colleges and universities usually offer computer-related degrees which combine education in a major field with computer courses. If, for example, you wanted to become a systems analyst and work primarily in business, you would probably study business administration with computer science courses included as part of your course of study. If, however, you wanted to become a systems analyst for scientific research in computers, you would most likely have computer science as your major field.

For whatever the specialized computer career you may want to pursue, college and university admission offices are usually helpful in supplying you with catalogs, pamphlets and other information about their computer science courses. High school counselors are trained to aid students in choosing the right college and the proper college courses for a particular career. You can begin now to explore the paths to a college education open to you by consulting these sources.

The care and feeding of the computer is the responsibility of the computer operator. Here at the console of an IBM 360/125 he and a company manager examine a printout showing the runs made the previous night.

FINDING THE RIGHT JOB FOR YOU

Assuming that you have gained enough computer training to look for a job, finding the job which is just right for you depends somewhat on what the job may offer you in the terms of salary, security, and the chance to grow and advance. You can learn to judge the quality of a job by comparing specific jobs in different organizations. It is also very helpful to contact people who are already involved in the kind of job you are interested in. They may be able to give you some added insights into the position and can offer you the benefit of their experience with the job and the organization.

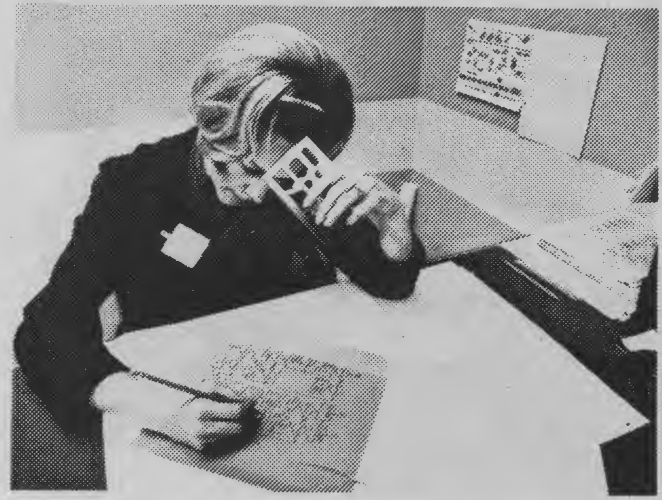
Considering the Opportunities

You would probably want to ask certain questions about opportunities and future advancement, regardless of the career path you choose. You might want to know how difficult it is to begin in a particular job, what the pay is, how secure the job will be, and what opportunities for advancement it offers.

For several years the demand for people in data processing will greatly exceed the available supply. Wherever demand is high, salaries are high. The high salary range in data processing will continue for years as the need for qualified people remains.

Security for the future is also high when demand is high. In data processing, however, security does relate a great deal to your ambition to learn and keep up to date with a rapidly changing industry. There is little need in data processing for people who wish to remain in the same job performing the same tasks year after year.

Opportunities for advancement in computer-related careers are unmatched anywhere. Again, the great demand for people is the reason. Because of a great need for trained personnel, advancement is more rapid than in any other field. New employees and long-time employees tend to



A programmer determines that the most efficient way to approach a problem, generally making use of a flowchart. Following this, the programmer writes the problem in computer language and tests it on the computer. (Photo IBM)

have equal opportunity because everyone in the computer center is constantly learning or training for new tasks.

Whatever computer skills or training you decide to gain in the next few years, there will be well paid and rapidly advancing careers open to you.

Meeting People in the Field

You have, so far, seen some possible ways to explore your interests and to look into some available careers and opportunities. You can read about how computers areused in a vareity of organizations, and you can also read books and pamphlets published by individual firms or by scientific research laboratories or educational groups. Enough has been written about careers in computer science and data processing to give you a good overall view of the world of computers.

Once you know what careers in computers are of interest to you, you can take another step toward understanding the computer world. Probably one of the best ways to learn about a specific job in an organization of your choice is to talk to the people who work in that environment. The people who have already gained some experience as computer specialists are generally most helpful. They will usually understand your interest and ambition to learn about the jobs they are doing.

Find out which businesses or industries, scientific organizations or school districts in your area have a computer center. You may even have already decided that you are interested in only one of these areas. If you have decided you want to talk to someone working with computers in business, locate the local banks, insurance company, airline, or organization of your choice where you can meet someone in the computer center. Many computer centers for scientific research are organized by government agencies. You can contact your state government information office for details about their research centers. Schools, of course, are listed in your telephone directory. Your own school district may have a computer installation or may have access to one.

Once you have located a computer center, call and ask to speak with the person in the position you want to know about. In most cases, you may not only talk over your career plans and nature of the job with that person, but may be offered a tour of the computer center as well.

You may even have a personal friend who has experience with computers who can tell you what his job is like. Whatever way you go about meeting the people in data processing, you can learn more about the computer career you decide upon by getting some firsthand impressions.

The extent to which you explore these computer careers is up to you. Remember, however, that learning and planning for your career can be the most important step you take.

RABBIT CHASE

by Ted C. Park
Pacific Union College
Angwin, California

DESCRIPTION

Seemingly, the purpose of this game is to chase-down and catch a rabbit. Now this rabbit is an elusive little devil — it can hop randomly in any direction. You can run at least as fast as the rabbit, maybe even faster (the computer will decide). You must get within 20 units of the rabbit to be able to catch him. Before each hop, the computer will print out your position, the rabbit's position, the direction the rabbit is going to jump, and your closest approach on the last hop. You are to tell the computer which direction you wish to run. All coordinates and directions are as a geometer would mark them on a standard Cartesian Coordinate System.

Really, the purpose of the game is to give you practice in using and visualizing an x-y coordinate plane.

USING THE PROGRAM

1. This program will run on most versions of BASIC, if your computer does not like it, convert it.
2. Consider the output and try to run the right direction.
3. Try to do all the figuring in your head. Using scratch paper is considered to be cheating (except for maybe the first time you play).

SUGGESTED MODIFICATIONS

1. Change the program so that you can choose your own speed.
2. The game is much more challenging when the "capture distance" can be varied. A distance of 50 units is a cinch, 15 units may make you wish for scratch paper, 5 units will require you to use a protracter and graph paper.
3. See if you can invent a way to extend this game to 3 dimensions! 4 dimensions!!! etc.!!!!!!
4. You might try limiting the total number of hops and/or having the computer give hints when requested.
5. If your BASIC supports a "print using" type of statement try rewriting the output section in a more clever manner.

SAMPLE RUN

```
SPEEDS (UNITS/HOP):
RABBIT - 140   YOU - 420

HOP#: 0001      DISTANCE TO RABBIT: 0445     CLOSEST APPROACH: 0445
RABBIT ---      POSITION: (-0180,-0407)      AND DIRECTION: 0153
YOU ------      POSITION: (+0000,+0000)      AND DIRECTION:?220

HOP#: 0002      DISTANCE TO RABBIT: 0075     CLOSEST APPROACH: 0075
RABBIT ---      POSITION: (-0305,-0343)      AND DIRECTION: 0189
YOU ------      POSITION: (-0322,-0270)      AND DIRECTION:?270

***********
* GOT YA *
***********
```

LISTING

```
CHASE

100  REM ( 'T' IS THE SQUARE OF THE CAPTURE DISTANCE )
105  LET T=400
110  REM
115  REM --  INITIALIZE VELOCITIES AND POSITIONS
120  REM
125  LET V1=INT(RND(0)*10+.5)*10+50
130  LET V2=(INT(RND(0)*2+.5)+1)*V1
135  LET X1=(INT(RND(0)*400)+100)*SGN(RND(0)-.5)
140  LET Y1=(INT(RND(0)*400)+100)*SGN(RND(0)-.5)
145  IF Y1=0 OR X1=0 THEN 135
150  LET X2=0
155  LET Y2=0
160  PRINT "SPEEDS (UNITS/HOP):"
165  PRINT "RABBIT -";V1,"YOU -";V2
170  PRINT
175  PRINT
180  PRINT
185  LET C=(X2-X1)↑2+(Y2-Y1)↑2
190  LET P1=3.14159/180
195  LET H=1
200  REM
205  REM --  PRINT OUT
210  REM
215  LET D1=INT(RND(0)*359)
220  PRINT "HOP#: ";
225  LET Z=H
230  GOSUB 545
235  PRINT "     DISTANCE TO RABBIT: ";
240  LET Z=SQR((X2-X1)↑2+(Y2-Y1)↑2)
245  GOSUB 545
250  PRINT "     CLOSEST APPROACH: ";
255  LET Z=SQR(C)
260  GOSUB 545
265  PRINT
270  PRINT "RABBIT ---      POSITION: (";
275  LET Z=X1
280  GOSUB 520
285  PRINT ",";
290  LET Z=Y1
295  GOSUB 520
300  PRINT ")       AND DIRECTION: ";
305  LET Z=D1
310  GOSUB 545
315  PRINT
320  PRINT "YOU ------      POSITION: (";
325  LET Z=X2
330  GOSUB 520
335  PRINT ",";
340  LET Z=Y2
345  GOSUB 520
350  PRINT ")       AND DIRECTION:";
355  INPUT D2
360  IF D2<0 OR D2 >= 360 THEN 355
365  PRINT
370  PRINT
375  REM
380  REM --  COMPUTE PATHS AND SEE IF THEY INTERSECT
385  REM
390  LET X3=V1*COS(D1*P1)/100
395  LET Y3=V1*SIN(D1*P1)/100
400  LET X4=V2*COS(D2*P1)/100
405  LET Y4=V2*SIN(D2*P1)/100
410  LET C=(X2-X1)↑2+(Y2-Y1)↑2
415  FOR I=1 TO 100
420  LET X1=X1+X3
425  LET Y1=Y1+Y3
430  LET X2=X2+X4
435  LET Y2=Y2+Y4
440  LET C=C MIN (X2-X1)↑2+(Y2-Y1)↑2
445  NEXT I
450  LET H=H+1
455  IF C>T THEN 215
460  PRINT
465  PRINT
470  PRINT "***********"
475  PRINT "* GOT YA *"
480  PRINT "***********"
485  PRINT
490  PRINT
495  PRINT
500  STOP
505  REM
510  REM --  CONVERTS NUMBERS TO STRINGS FOR CLEANER OUTPUT
515  REM
520  IF Z<0 THEN 535
525  PRINT "+";
530  GOTO 545
535  PRINT "-";
540  LET Z=-Z
545  LET Z=INT(Z+.5)
550  DIM S$[10]
555  LET S$="0123456789"
560  FOR I=1 TO 4
565  LET W=INT(Z/10↑(4-I))
570  PRINT S$[W+1,W+1];
575  LET Z=Z-W*10↑(4-I)
580  NEXT I
585  RETURN
590  END
```

It's fun in the right group . . . a small group. Time is the problem, the sales rep is the friend, the establishment is the enemy . . .

The Quality of Life

by Milt Stone, Contributing Editor

"Consider the difference . . . The computer in its air-conditioned, dust-free, sanctum sanctorum . . . Next door, the programmer, in ghetto-like squalor, with printouts on the coatrack and cards on the floor . . ." ROBERT PROPST[1]

Garry Huggins is not a programmer but his desk, chair and feet are in constant contact with asphalt tile. The walls of his office stretch only half way to the ceiling but the carpeting beneath his System 3 covers the entire computer room. And he's having a ball (at E. B. Wiggins, Inc., in Los Angeles).

He was trained for this business. College work in business and accounting. Credit job with a steel distribution outfit. Branch manager for a wholesale liquor dealer. And then the big change. When the liquor dealer was faced with a conversion from a 1401 to a 1440, the installation manager knew how to program—so he became the programmer; Garry was a proven office manager—so he became the installation manager.

"But I knew I could be snowed," Garry says, "so I learned programming and I found that I had an aptitude for it." He got his kicks then, and still gets them, from "outguessing the machine."

Eventually he left to work, for less money, for a large insurance company—with a big stable of computers from large to small. "I had my own little corner, mortgage loan accounting and forecasting policy revenue. Systems work and programming. I got a lot of education but it was no fun."

Finally, the chance to run his own show, again. He now works for a $15 million, engineering-oriented, family-owned business—IBM, having found him, blessed him—the family wants them to approve—and he's built his operation from a monthly hardware bill of $2K to $6.5K in just two years. At the same time, he's cut his staff from six operators to one and from two programmers to one. "We had a three-shift operation producing nothing—now we have a one-shift operation and we'd be out of business without it."

What next? "Well, I'm happy here and there's a lot to be done. But ultimately I'd like to work for the top guy somewhere and have all administrative services reporting to me."

Garry Huggins' story is a good case in point. For a very large number of computer people: getting involved in the business was an accident; "outguessing the machine" is the fun part of the game; and being in a small, closely knit group with plenty of access to the computer makes all of the difference between a good work situation and a bad one.

In Garry's case, being with a small installation, and a small company, means "having contact with all facets of the company"—something that just didn't happen at the big insurance company. And for the honcho in a small installation the job is a different one. Superhoncho is primarily an evaluator of the people under him who give him technical advice—only secondarily an evaluator of the advice. The small installation honcho is a working foreman; *he* is the top techni-

. . . a good number of people in the computer business are where they are because IBM put them there.

1. President, Herman Miller Research Corp.

cian—Huggins still programs and likes it—and his is the most important voice in a technical decision (after the vendor's, of course). Speaking of the vendor: a good number of people in the computer business are where they are because IBM put them there.

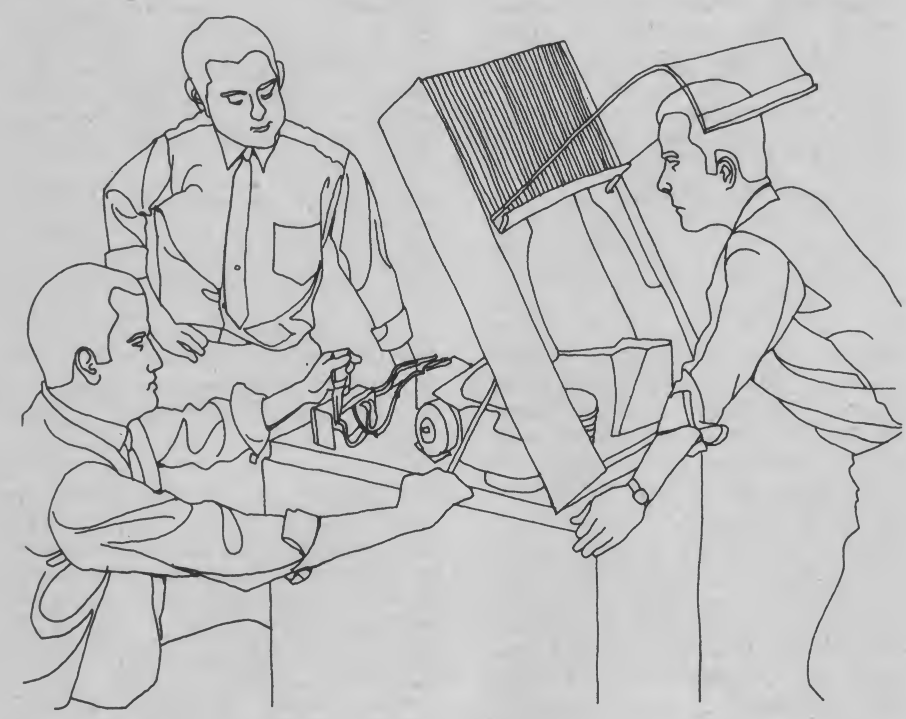
When the time comes for Huggins to move on (and this is understood to be inevitable by his employer), "IBM will help me," he says. "That sales rep can really do the manager some good." On that note he left for lunch—with the rep.

St. Louis, McDonnell Douglas Automation Co. and Ted Bellan are miles and light years away from Los Angeles, Wiggins and Huggins—and Bellan, vp-computer services, is the superhoncho exception who proves the rule. (He, the boss, may be the top technician—in a big shop.) He has two degrees in engineering, came up the ladder on the scientific programming side, has been heavily involved with the operation of hardware from the CPC in 1952 to the 360/195 in 1972—the lifetime of IBM computing. He calls his staff of 1,000 computer types the "engineering and manufacturing" segment of the automation company.

Bellan knows his business and therefore can be direct with the people who work with him—vital to morale, anywhere. He demands respect as a guy who knows so much about this discipline that his leadership is contagious. He talks about a familiar scene—repeated at other companies, too—the IBM man, the telephone man, and the independent peripheral man listening to Bellan say, "Nobody's going to leave this goddam office until you guys agree that everything's OK." You get the feeling that in this case—in the presence of this savvy guy—they felt sheepish and uneasy, especially if they were trying to get out of a responsibility.

Bellan says, "A company thinks highly of manufacturing, marketing, finance, because they produce profits. But data processing doesn't. It's considered a necessary expense and treated as such in many cases."

Although Ted Bellan was never unhappy with his lot, his outlook is a lot brighter now that he can combine computing with profit making. As J. Don Reisser, a Los Angeles headhunter, says: "Like it or not computer people are staff people. One of the comforts of staff people is that they don't have to put themselves on the line (for achieving results). One of the hazards of a staff job, however, is that that's where the cuts are made when times are bad. I tell my young son, 'Get a line job. You learn more. You earn more. You have more fun.' " Ted Bellan is having more fun in McDonnell Douglas Automation than he had as staff to the bird builders.



In New Orleans, at the Hibernia National Bank, David Burns is a 24-year-old computer operator who comes on like a banker—conservative dark suit, white shirt, neatly-trimmed hair with short sideburns. Four years ago he was working his way up the ladder in the Winn-Dixie Stores chain, taking junior college courses in business administration at night. A friend who worked for the bank suggested that he apply for a computer room job. In seven months he graduated from the "menial tasks" to an operator's job. He's now a day-shift operator. In the four years he's doubled his base salary. (The Hibernia is an NCR shop.)

He likes his job, thinks it's a good job, and not really too difficult, though there's a lot to learn. He says his father is proud of his son's job as a computer operator—"the coming thing"—and Burns is proud of it himself.

As for advancement, Burns is uncertain whether he should become a programmer or a supervisor of operators. He seems to lean toward the former, and is a bit regretful that he turned down an opportunity to take the programming aptitude test the last time the bank administered it a year and a half ago. He'll take it next time.

Before he rises higher, he needs a deeper knowledge of the computer and software. He'd like to understand more about programming in order to understand his work better. He now knows how to run nearly all jobs, though. He doesn't feel courses are a help—he says he learned far more from on-the-job-training than he would have from books.

"Jack Younger" works as an "information systems designer" for a large company headquartered in a small city in the boondocks. You'll meet his duplicate frequently, and this is how life looks to him:

How do you spend your time? "Maintaining current software packages, We are not developing any new packages until we get the ones we've been working on running. For example, we have an inventory program that has been in development for six years and still doesn't run very well,

The Quality of Life

and a payroll program that isn't running the way it ought to after five years. Management froze the programming section and said that no new packages would be started until these were running. Everyone is in limbo, and the morale is low, but people can't job-shop now—there just aren't any jobs to be found."

What went wrong? "I don't know. When I came here several years ago, the inventory package was out of control even then. It hadn't been planned right from the beginning, but something had to be up and running. We started patching it. I thought we should have thrown the thing out and started over from scratch, but the company figures any program is better than none at all."

How good are your managers? "I've had one good manager here—out of six. He was interested in what he was doing, but he was interested in what I was doing, too. He would get the job done even if it meant sticking his neck out. The typical manager is just concerned with keeping his position. This guy was a real-time problem solver. I could walk into his office and give him a problem, and he would jump on the phone and solve it *right then.* But he was wasted on this company, and he moved on."

What's fun or worthwhile in your job? "Nothing's fun here. It's all bad." What looks like success to you? "My own software company, and I'm starting to work toward that. Nothing big —just developing general packages."

One theme running through all these interviews is clear: programming is at the heart of computermanship. Computerman must be aware of what good programming is if he is to get the technical part of his job done. If time spent in the job of programming—no matter how short the time is—is one of the

"I've had one good manager here—out of six . . ."

necessary steps for Computerman, then a knowledge of the attributes associated with the occupation of programming is useful. It's especially illuminating when used as a frame of reference for examining—as we will shortly—what the grunts and honchos say are their plans for the future and what they say are the obstacles in their way.

Based on his continuing research into the matter, Dr. Edward M. Cross, Old Dominion University, sees the programmer as "a loner, an individual who wants to avoid confrontation, avoid being directed, is willing to do without much social interaction on his job, does not have an interest in social service—has no apparent desire to enter into the aggressive, competitive, confrontation-laden situation that is associated with line managerships. He is a 'staff' man—but a staff man working very much in isolation."

Cross elaborates on this mini-portrait. No surprise: programmers tend to stick to a job even when not particularly interested in it. No surprise: they use step-by-step methods for processing information (as opposed to intuitive, impressionistic approaches). No surprise: they are not motivated by Theory X management or by rewards —accomplishment, getting the job done, is the spur. (Marion Bell, Programmatics, Inc., Los Angeles, on writing assemblers for minicomputers: "It's really neat when you deliver it and the customer starts to run it. It's fun when the pieces all come together and they fit. One thing—they only take

The Order of Battle

. . . in the trenches

In its battle to stay alive in a sometimes unfriendly environment, the data processing activity depends on the efforts of a familiar troika—marketing, engineering, manufacturing. Marketing does the product planning and sells its efforts. Technicians call this systems design. Engineering draws up the plans—programming. And manufacturing—operations—delivers the goods.

THE ANALYST LEADS

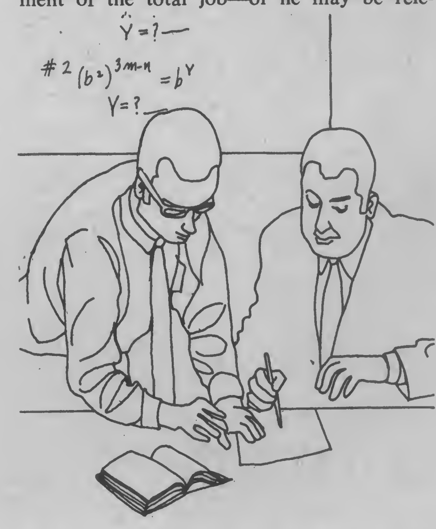
Grunts who design systems are called *analysts, designers, procedures people,* and *programmer/analysts* (if they do a cradle-to-grave job which includes writing the computer programs specified in their system design).

Given a project—for example, satisfying the requirement to handle customer orders from receipt of order through shipment of merchandise to final dispatch of an invoice to the customer—the systems analyst is expected to specify the flow of information that will do the job. His design must be workable, efficient, effective and politically acceptable. He is expected to work with the ultimate users of the system—the customers—and to incorporate their ideas and/or firm demands into his design. If he is creative and courageous, he proposes and presses for the acceptance of his own idea of the best way to get the job done. Thus, he must be both a good listener and an accomplished salesman.

ENTER THE PROGRAMMER

Applications programmers transform the individual computer programs visualized by the systems analyst into reality. The odds are good that a "lead" or "senior" programmer will second guess the analyst by restating some of the program functions specified—or by recombining the programs specified and resegmenting the whole into a different statement of individual programs. Why? Because it will be "better for the customer" or "more efficient" or because programmers are creative, too.

A "junior" programmer will probably be limited in his work assignments to writing code from flow charts produced by a "master"—probably for only a small segment of the total job—or he may be relegated to maintenance (the never-ending task of correcting errors and making minor changes in existing programs). In any event, the programmer is expected to see the program through its final checkout with both test data and live data, and he is expected to prepare adequate instructions for the operations staff. He, too, ideally is both a good listener and articulate.



The *programmer/analyst* does the whole job from talking to the customers to writing the operating instructions for the individual programs. His scope ranges from consideration of the grand concept at a user level to the detection of an obscure bug in a program. Often, the programmer/analyst claims that he has less need for communication skill because he can conveniently carry the details of the whole system in his head. Listen to whom? Talk to whom?

In general, analysts and programmers choose between two polarized environments and work in only one of them—with scientists and engineers or with accountants, production people and marketeers. The choice has something to do with the computer skills required. But it is much more likely to be based on educational background and the ability to cope with the special vocabulary of the environment.

In general, too, applications programmers today do not communicate directly with the computer. They live within a set of rules for programming which permits them to communicate with an operating system (or a data management system under the control of the operating system), which in turn communicates with the computer. More and more applications programming has become confined to setting up files and manipulating transaction data according to the stylized rules of a programming language—and talking to an operating system with the help of another code book.

What's different? For a starter, even minicomputers today have more capability than many of the earlier computers. The ability to squeeze the most out of very limited resources (by taking advantage of an intimate knowledge of the machine) is not very important any more. In addition, the jobs of automating the scheduling of the computer, three or four months to write and that's good. Other projects go on and on and you never see anything happen.")

No surprise: they may work irregular hours—frequently by choice. Surprise: they have a preference for stability and security as opposed to work which is irregular, challenging, dangerous, or otherwise exciting. Big surprise: programmers have very little desire to help other people as part of their jobs. Cross says that the data processing job is somehow devoid of far-reaching social impact (some would disagree).

Almost all of the grunts who contributed their thoughts to this series are, were, or hope to be involved (at some career stage) with programming. Five out of six see edp as their best and preferred career field, now and in the future. And almost every one who wants to stay in the field *sees himself as some sort of manager in the future*. There are exceptions.

... almost every one who wants to stay in the field sees himself as some sort of manager in the future.

"I don't want to be a manager because management is selling" (systems analyst). "I have managed two multiple-man projects and learned that I am not tough enough to want to do that again. I enjoy managing and being responsible to myself." (Marion Bell)

But the itch to be the leader is far more typical—even on the female side of the house. "Alma Bond" (SE-type IBM-er under wraps) would like to try management, specifically a supervisory job with eight to ten people under her. "I'm not sure I can handle it but if I find I can, I want to go as high as I can go—definitely within IBM." If she finds that she can't hack it as a manager, "I want to become a consultant systems engineer. There's not as much prestige . . . but there's lots of opportunity for fun, lots of challenge and I like to be with customers." And at National Life Insurance Co., Montpelier, Vt., Marcel J. Marineau, a senior analyst/programmer, has an optimistic but realistic view of the future: "Most positions on the management level require considerably more education than I have at the present time . . . but I can foresee that a position as a project leader over a number of programmers or systems people is well within my range."

But when you've had it, you've really had it. After nine years in the field and thirteen jobs, one programmer wrote, "My best career path is O-U-T — O-F — T-H-E — C-O-M-P-A-N-Y and O-U-T — O-F — T-H-E — F-I-E-L-D. I'm fed up."

A more reasonable—but less decisive—tone is adopted by a potential midwestern drop-out. His credentials: a BA in math, some graduate work, three jobs and 13 years in edp. His bag: operating systems, their care and tailoring. "My career advancement depends on my willingness to play the

---

making the best possible use of its resources, and, in fact, conventionally accessing each of its components have been taken over by the system software—the operating system—and by the *systems programmers* (a relatively small group). For many programmers, direct interaction with the machine was the fun part of the job.

AND SUPERPROGRAMMER

Systems programmers come in three flavors. The *developers* conceive and write the programs which comprise an operating system. Most of them work for computer manufacturers, some for software companies. Periodically, these companies release new versions of their operating systems. The *maintainers*, who work for the users, preside over the replacement of Version 12 with lucky Version 13 ("new and improved"). The introduction of each improved version can be counted on to cause a number of hitherto smoothly performing application programs to blow up. This adds another dimension to the job of the maintainers. They must seek out the reasons why the programs have blown, asking the manufacturer's representative to tell them what changes were made in the operating system. Then the maintainers can specify what changes in the application programs have now become mandatory. Finally, the *tuners*, who also work for the users, have the task of balancing the variable elements of the operating system so that throughput is maximized: they get rid of as much operating system garbage as possible—without the roof falling in—so the installation can get some work done.

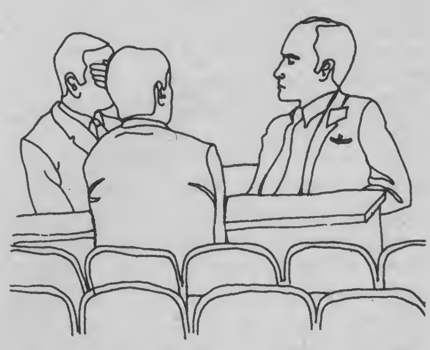
ALSO SUPERGRUNT

Finally, and really in the trenches, are the *computer operators*, the supergrunts. James A. Campise, Houston computer consultant, paints this picture of them: "Look into any busy information processing center and you will see several young men and women pushing buttons, changing magnetic tapes, flicking through punched cards and in other ways supervising the operations of the computer . . . When the control panel lights indicate that the machine has stopped for some reason, the console operator must investigate and correct the stoppage . . . He serves an apprenticeship . . . inserting punched cards in card readers and punches, inserting forms in printers, mounting reels of magnetic tape on tape drives and generally readying peripheral devices for operation . . . Many computer operators have successfully advanced to positions as supervisors of operations, programmers and computer center managers."*

The operators belong to the production staff, the manufacturing department, of the "information business." They receive the raw data flowing into the "factory." Regardless of whether it arrived at the right time or the wrong time, they must push that data through the complex of machines and programs to produce a finished product, timely information. As is true of production workers in other sophisticated factory environments, operators are expected to be able to "cope" when things go wrong—which is frequently.

... and at the club

Honchos are variously called *manager, director* or *vice president* of *data processing, systems, information systems, management information systems* or *management systems*. These are the guys who seemingly have it made in their chosen field. At any rate, some of them are firmly dug in. It has been estimated that one-third of the honchos have been with their present firm for nine or more years—and one-half have been around more than five years. Does this mean that the data processing honcho is finally stabilizing his own career planning and achieving a modest measure of security? Or does it mean that there's trouble in Computer City—that Computerman is losing the struggle for the honcho hat in more and more companies?



Leonard W. Snodgrass has been in charge of data processing at General Tel of California since the early 60's. As vice president and controller he was also in charge of five other functional areas. Until last July. At that time, Len Snodgrass explains, "Although my prime responsibility over the past, more than 30 years, has been financial—executive management felt that my knowledge of the operation of a telephone utility . . . financial background . . . administrative experience in edp particularly qualified me for a new assignment: vice president—data systems. This gives you a strong indication of top management's interest in data processing functions." And of top management's specification for honchos, too.

*AFIPS, Computer Careers, reprinted 1970.

corporate game. Ideally, I'd like to own a small, profitable business (entirely outside of data processing) and work occasionally as a free-lance in data processing. Practically, if I can achieve the stature of an 'expert,' I'd be better off elsewhere in the company . . . if not, I'd better stay where I am."

The honchos have a different view of their future in edp. There are traces of optimism, aggressiveness, cynicism, disillusionment, realism. Three out of five are committed to continuing in the edp field—despite the fact that many of them feel that they really have no way to improve themselves in a career sense. A solid 40% are looking forward to either a spinoff situation—a new and better career which has nothing to do with the management of edp—or an escalation situation—in which the edp function is absorbed into a grander, more pervasive activity over which they have control.

An aggressive view of escalation: "Frankly, I don't see a career path until the edp organization *is* the rest of the company, serving the whole works and with a voice at the top. We have to put our arms around the whole animal and, in our embrace, make it hum. I suppose it sounds like supreme arrogance but the fact remains that we feel we know more about running the company than the very people who *are* running it—and I'm not convinced that at one point in time that won't be fully acknowledged."

R. W. Blaylock, vp—management information services, Plough, Inc., Memphis, talks about escalation in more restrained terms: "I visualize my career moving into general management with specific interest in data processing, long-range corporate planning and management practices (the ground rules by which a company governs its own interactions to insure that corporate goals are met)."

The honcho who wants to spin his career off into other fields and the honcho who plans to continue in edp both say, in effect, the same thing, "I'm trapped." But the onward-and-upward-in-edp man concludes, "This is the thing I know best; hopefully there's room to grow; but, even if there isn't, I'm staying." An eastern manager of scientific computing put it this way: "I would give anything to get out of this disgusting and degrading staff job and into a line job. (He had previously labeled the work 'exhilarating'—the selling 'infuriating and discouraging'.) But at my level and salary there's no line career path open. (He's 50+ and makes $30K.) I am condemned."

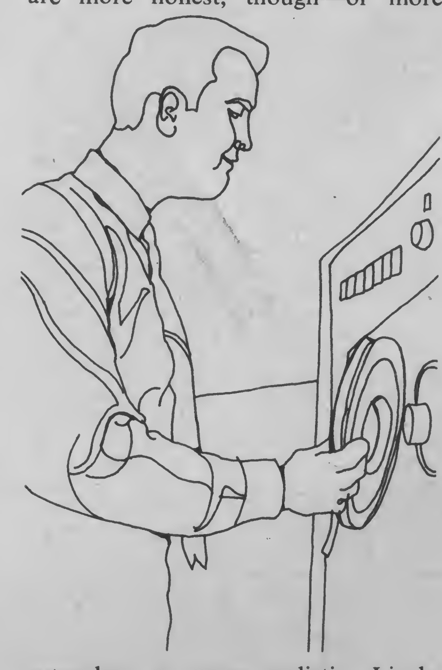
The spinoff simply says, "It's been great but now I'm trapped—I want to get out of edp and into the green world outside . . . "

In predictable ways some elements of a discussion of obstacles to success are the same inside or outside the computer world. The blacks feel put upon —lack of opportunity, both to get in and to go up. Some of their non-black associates, in turn, feel that all too often blackness is blatantly used as a substitute for performance.

Some of the women feel put upon. Most don't. A few are not at all hesitant to use their womanhood as a weapon—to the delight, irritation, or indifference of non-women. Nothing new there, either.

. . . 40% are looking forward to . . . a new and better career which has nothing to do with the management of edp . . . or in which the edp function is absorbed into a grander, more pervasive activity over which they have control.

Practically, too, both honchos and grunts are reluctant to look inward in their discussion of obstacles. Grunts are more honest, though—or more outspoken or more realistic. Lively Marion Bell again: "I'm the biggest obstacle in the way of my professional development and career advancement. I am a social creature and relish my private time away from the office." (After five jobs and almost 13 years in the business.)



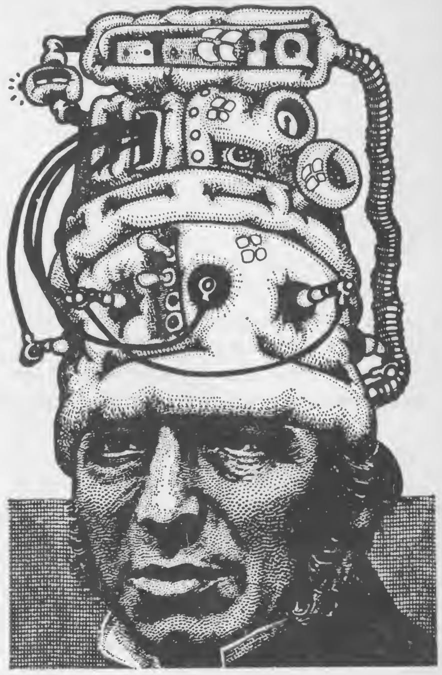
Time—private or not—is the big obstacle in the way of self-development to grunt and honcho alike. Listen to Tom Southard, an outstanding manager of an outstanding installation (Westinghouse Air Brake Co., just outside of Pittsburgh): "Ours is a small, highly professional shop and I really have to be all things to all men. I talk to advertising agents, work up and live with budgets, lecture, provide a shop quality control function. I do everything around here, it seems. I haul the water, bake the bread, make the beds, sharpen the pencils, sweep the floors. If I could I would dedicate one whole of myself to administrative things and another whole of myself to professional things. My wife says, and she's probably right, that I have exchanged the family for my work."

Tom is a Phi Beta Kappa, a graduate summa cum laude from Kenyon College, a Fulbright Fellow, and a veteran of 17 years in edp. If time is a problem to him, consider the plight of Joseph J. G. Brooks, programmer/analyst by day (UFC Systems, Philadelphia), Villanova student at night, 30+ years old, and overwhelmingly involved in the process of bootstrapping himself up. Brooks started as an IBM customer engineer on unit record equipment. "I work full time . . . go to college full time at night . . . three hours per day for travel . . . six for sleep . . . the whole big weekend for studying, family life and maybe a little basketball. I do not have the free time I would like to improve my technical skills."

The grunts don't have a clear picture of what obstacles there are along their climb to the coveted managerships. They lack leaders, the economy is bad, the installation is too small, there's too much politicking, they're too old, they're too young, they lack education —it's a mixed bag. The honchos, however, have a clear picture of the "doomsday defense" standing between them and their rightful place in the sun. The enemy is the "organization establishment."

The descriptions of the problem were various. The explanations of the enemy's thinking ran the usual course from ignorance to fear to envy. But it all came out the same—the real problem for the dp honcho is "thoroughly entrenched top management, paying lip service to the new technology when all the time they're massively afflicted with the accountant's syndrome. They don't understand that the name of the game isn't to save money—it's to make it."

Test For System Analysts



INSTRUCTIONS: Read each question carefully. Answer all questions. Time limit: 4 hours. Begin immediately.

HISTORY: Describe the history of the papacy from its origins to the present day, concentrating especially, but not exclusively, on its social, political, economic, religious, and philosophical impact on Europe, Asia, America, and Africa. Be brief, concise, and specific.

MEDICINE: You have been provided with a razor blade, a piece of gauze, and a bottle of Scotch. Remove your appendix. Do not suture until your work has been inspected. You have fifteen minutes.

PUBLIC SPEAKING: 2500 riot-crazed aborigines are storming the classroom. Calm them. You may use any ancient language except Latin or Greek.

BIOLOGY: Create Life. Estimate the differences in subsequent human culture if this form of life had developed 500 million years earlier, with special attention to its probable effect on the English parliamentary system. Prove your thesis.

MUSIC: Write a piano concerto. Orchestrate and perform it with flute and drum. You will find a piano under your seat.

PSYCHOLOGY: Based on your knowledge of their works, evaluate the emotional stability, degree of adjustment, and repressed frustrations of each of the following: Alexander of Aphrodisias, Rameses II, Gregory of Nicia, Hammurabi. Support your evaluation with quotations from each man's work, making appropriate references. It is not necessary to translate.

SOCIOLOGY: Estimate the sociological problems which might accompany the end of the world. Construct an experiment to test your theory.

ENGINEERING: The disassembled parts of a high-powered rifle have been placed on your desk. You will also find an instruction manual, printed in Swahili. In 10 minutes a hungry Bengal tiger will be admitted to the room. Take whatever action you feel appropriate. Be prepared to justify your decision.

ECONOMICS: Develop a realistic plan for refinancing the national debt. Trace the possible effects of your plan in the following areas: Cubism, the Donatist controversy, the wave theory of light. Outline a method for preventing these effects. Criticize this method from all possible points of view. Point out the deficiencies in your point of view, as demonstrated in your answer to the last question.

POLITICAL SCIENCE: There is a red telephone on the desk beside you. Start World War III. Report at length on its socio-political effects if any.

EPISTEMOLOGY: Take a position for or against truth. Prove the validity of your stand.

PHYSICS: Explain the nature of matter. Include in your answer an evaluation of the impact of the development of mathematics on science.

PHILOSOPHY: Sketch the development of human thought, estimate its significance. Compare with the development of any other kind of thought.

GENERAL KNOWLEDGE: Describe in detail. Be objective and specific.

creative c

★ a lively new non-profit magazine of edu

CREATIVE COMPUTING
P. O. Box 789-M
Morristown, N. J. 07960

computing

educational and recreational computing ★

"The Snail" was produced on a CalComp 770/702 Plotter. It took 25 minutes of computer time and 4½ hours of plotter time. Courtesy California Computer Products, Inc.

Squaresville Results

The Squaresville problem in the Nov-Dec 74 issue asked you to arrange two of each of the digits 0 to 9 to form a 20-digit number. The number then was scored as follows: for every two consecutive digits forming a perfect square, score two points. For each three consecutive digits forming a perfect square, score three points. And so on. What is the maximum number of points you can score?

We're a bit embarrassed that the winner is our own Reviews Editor, Lynn D. Yarbrough. He comments that he has no idea of a reasonable way to attack the problem with a computer, but using a hand-held scientific calculator and some ingenuity seemed to yield a good solution. The number Lynn found, which scored 100 points, is:

49134681827562500379	Square Root	Points
49	7	2
134681827562500	11605250	15
1346818275625	1160525	13
81	9	2
1827562500	42750	10
18275625	4275	8
27562500	5250	8
275625	525	6
7562500	2750	7
75625	275	5
562500	750	6
5625	75	4
62500	250	5
625	25	3
2500	50	4
25	5	2
		100

Lynn comments further, "the longest single square I have found with no tripled digits is $9987338075625 = 3160275^2$, which can be neatly combined with $25210441 = 5021^2$ into the sequence 99873380756252104416, but its score is not nearly so high (40)".

For those readers wishing to continue with Squaresville, there is a number with a score of 106 points. I leave it as a challenge for you to find it. — DHA

Call for Games

Do you have a favorite computer game? Why not share it with the readers of *Creative Computing*? Here's how to submit a game to us for publication:

1. Programs must be in BASIC, complete, debugged and with complete playing instructions for a beginning user. Watch English, grammar and spelling.
2. Please send us:
   A. Program listing (and paper tape, if possible)
   B. Two sample runs the way an AVERAGE person would play
   C. Brief verbal description of game, suggestions for modifications, related activities, etc.
   D. Description of any unusual computer or language features used. (Please try to use a standard version of BASIC).
   E. Your name, address, telephone, age, school or company affiliation, computer system used, original source of program (if not you).
3. Listing and runs must be on *white, unlined* paper. If you have lined paper, turn it around to the unlined side. We cannot publish material on yellow, pink, blue, or gray paper. Xerox, Ditto, or other copies are also unacceptable.
4. Listing and runs must be done with a *fresh black* ribbon. Not purple or blue and especially not a used ribbon. The Teletype ball must be clean, in good adjustment, and produce crisp copy. If necessary, clean the ball with a typewriter cleaner or stiff toothbrush.
5. If possible, submit a paper tape of the program. *Be sure* to wrap oiled paper tape (from Teletypes) in kitchen plastic wrap when you mail it. Otherwise the oil seeps out and smears the output.
6. Accompanying program descriptions should be typed double space.
7. If you want an acknowledgment that your contribution was received, include a stamped self-addressed envelope.
8. By submitting a program, you are giving *Creative Computing* the right to publish, reprint, distribute, or use your program in any other way. You will, of course, always be credited as the author.

HINT: Creative, original, interesting games are more likely to be published than new versions of old games. For example, we will absolutely *not* publish the following games (they are just too worn): Blackjack, Calendars, Craps, Football, Horse Race, Nim, Slot Machine, and Stock Market.

Contest! Win$$

CONTEST RULES

Contests are open only to subscribers.

All contest entries must include the following information: your name, home address, home phone, age, school or company affiliation, contest problem number and your answer(s). Problems that call for the use of a computer must also include program listing, output, paper tape or card deck. Entries should be in BASIC or FORTRAN and should not use unique features of a particular compiler. If you are a high school student, you should also include the name, home address and phone number of your teacher.

Entries not including the above information will be automatically disqualified. Entries cannot be acknowledged.

The winner will receive a two-year subscription extension (or gift subscription) to *Creative Computing*. Second and third place winners will receive a one-year subscription extension (or gift subscription). Winners will be announced in *Creative Computing* two issues from now.

The winner, along with two associates (teacher, friends, etc.) will form a committee to judge the next contest. Contest entries may be judged on computer technique, uniqueness of the program, form of the program or output, length, running time, etc. Decisions of this committee are final.

All correct entries (winners or not) will go into a year-end drawing for $25, $15, and $10 prizes.

CONTEST PROBLEM 3. MANHATTAN

We are told that the purchase price of Manhattan Island in 1626 or 1627 was $24. Assume the Indians invested their money on January 1, 1627 at 6% compounded annually. At that rate, the money would be worth a great deal by January 1, 1975. We know that if P dollars are invested at an interest rate of r (expressed as a decimal), the amount A the following year is given by:

$$A = P(1 + r) \qquad (1)$$

and the total amount after n years is given by:

$$A = P(1 + r)^n \qquad (2)$$

Write a program to calculate the value of the investment on January 1, 1975 to the nearest penny using two methods: (1) compute the amount year by year using Formula 1 and accumulate it, (2) compute the amount by Formula 2. In Method 1, round the result each year to the nearest cent. Your challenge, of course, is to maintain more significant digits than most computers are capable of handling.

You'll be interested to compare your answers with the 1974 assessed value of the land of some $6.4 billion.

CONTEST PROBLEM 4. CUBESVILLE

This contest problem does not require the use of a computer — you may use one, but you don't have to.

Arrange two of each of the digits 0 to 9 to form a 20-digit number. Your number may not begin with a zero. Score your number as follows:

For every two consecutive digits that form a perfect cube, score two points; for every three consecutive digits that form a perfect cube, score three points; and so on.

For example, if your number was 45864096859013312727, you would score two points for 64, four points for each for 4096, 6859 and 1331, and two points for each of the 27's . . . which would give you a total of 18 points. You may not count 01331 as a five-digit cube.

Games & Puzzles, Dept. 4
11 Tottenham Court Rd.
London W1A 4XF, England

CONTEST 1 RESULTS

Contest judged by:
Bill Stevens
Calif. State Univ.
Northridge, Calif.

We received several fine solutions. It was difficult to select one over the others. All used essentially the same approach, a straight forward testing of numbers from 1 to 10,000, with an output for those numbers that turned out to be digital invariants.

If you'll excuse a bit of propaganda we strongly favor simple, straight-forward programming. Quite often cute tricks and fancy routines backfire awkwardly. But on the other hand, some sophisticated thinking might be in order with a problem such as this. A brute force method would be time consuming if seven and eight digit numbers were being investigated.

As a winner, we must go with MELINDA HARP. She has written a fine program in BASIC for a HP2000F or a CDC 6400. We also appreciated her flowchart. Melinda is a student at Westside High School in Augusta, Georgia. Congratulations to Melinda.

As we mentioned earlier, there were other fine solutions submitted. We were impressed by the program submitted by Steven Epstein. He allowed for testing for invariants through 8 digits. Frank Pittenger provided a particularly nice form for his output. But we will go with the programs of J. Gabriel and Andy Astor as a tie for second place. We like their simplicity.

WINNER: MELINDA HARP
Westside High School, Augusta, GA

SECOND: J. Gabriel
Riverdale Country Day, Bronx, NY

Andy Astor
Scarsdale High School, NY

HONORABLE MENTION:

Richard Dolce
Carolyn Evans
Medical College of Georgia

Steven Epstein
Conant High School, IL

Paul Garmon
Wellesley High School, MA

Loretta Marzee
Oil Belt Vo-Tech School, AR

Frank Pillenger
Reynoldsburg High School, OH

D. Vollom
White Bear High School, MN

D. Zeigler
Vanguard HS, Waco, TX

CITALA

An Exploration of Instructional Computing in a Two-year College

David Howard
Bruce Corliss
Delta College
and
Karl L. Zinn
University of Michigan

This report on Project CITALA was adapted by Diann Bradarich from an article in the ON-LINE newsletter. A lengthier manuscript is available from Gene Arnold, Delta College, University Center, Michigan 48710 or Karl Zinn, Center for Research on Learning and Teaching, 109 E. Madison, Ann Arbor, Michigan 48104.

Realizing the potential impact of academic computing, Delta College in University Center, Michigan initiated an institutional study to investigate the use of the computer as an instructional tool in the community college. During the spring of 1973 a faculty seminar on instructional use of computers developed the nucleus for the Delta College staff team which conducted Project CITALA (Computers in Teaching and Learning Activities).

From the outset, the Project was largely faculty-oriented. Althouth the basic responsibility for the Project rested with the President, who provided general direction, the daily planning and development were the responsibility of the Team. All decisions regarding methods of procedure, institutions to be visited, consultants, and other details regarding the specific direction of the Project were made by the Team members.

The basic philosophy governing formation of the Team and its subsequent activity was inspired by the college's own *Institutional Profile* as of 1972. As a student-centered institution, Delta College bases future action on answers to the question: "What is best for the student?" The college seeks to provide educational services that prepare students for immediate employment and, certainly, experience with computers is becoming an increasingly valuable skill. Delta is committed to experimentation and research in areas which facilitate learning; thus, the decision to explore computing was a logical conclusion.

Domain for the Study

The Project Team concentrated on possible expansion of computer facilities as support tools and investigated feasible applications for Delta College. CITALA staff soon recognized that instructional uses of computers divide naturally into two areas – instruction *about* and instruction *with* computers. In instruction *about* computers, the machine is the subject of the study; it is not substituting for or replacing some other device, and it cannot be replaced itself. In instruction *with* the computer, however, the machine is a tool of instruction used when it offers some instructional advantage.

Delta's present equipment configuration is primarily designed for and used for instruction *about* computers and does not lend itself well to an expanded program of instruction *with* computers. Of course, instruction *about* computers is by far the greatest proportion of instructional use of the computer in education today, and the vocational training provided at Delta will continue to be considered. At the same time, however, exciting things are happening in the area of instruction *with* computers, and the activities of the Project Team were focused primarily in this direction.

Goals and Procedures

CITALA staff members engaged in extensive and varied activities directed towards the implementation of ten specific objectives: investigate many examples; visit key locations; collect information; collect materials organized by discipline; plan implementation; determine methods for faculty involvement; determine time sequence for expansion; design in-service programs; determine necessary expertise; and locate experts for advice. The major activities can be classified into several categories: identification of resources; use of consultants; trial computer use; and data collection from site visits.

Team members spent several days familiarizing themselves with project objectives and present facilities and programs at Delta. From the exchange of ideas emerged several valuable techniques designed to encourage effective operation as a team. A library file housing all material gathered on visits and through correspondence was maintained for the general information of all members. A checklist developed by the team assisted members in gathering pertinent data during visits to other colleges. Regularly scheduled meetings aided members in planning and sharing the analysis of their activities.

After the preliminaries of orientation, serious investigation began. Invitations to visit the campus were extended to consultants in instructional computing. Karl Zinn, who earlier directed the faculty seminar, visited Delta several times to help with planning, resources the preparation of reports. A team of experts from the University of Illinois spent two days at Delta demonstrating the PLATO terminal. In addition, conversations were held with a number of other experts from across the country, several of whom have since visited the Delta campus.

Through the cooperation of Project EXTEND (funded by Exxon Education Foundation) at the University of Michigan, a portable terminal was available on campus throughout the Project for use of computers at Michigan State University, the University of Michigan, and Dartmouth College. Team members and other interested persons gained experience in using the terminal and tested various educational programs that are stored in the computers at the three different institutions.

The last and perhaps most crucial activity focused on the collection of data from colleges visited by the Project Team. The highly worthwhile activity provided an opportunity to see new approaches in action as well as to establish contacts for long range developments in instructional computing.

Visits to Colleges Using Computers in Instruction

In order to select educational institutions for visitation, several criteria were established. First, the school had to be actively engaged in instructional computing; the Team wanted to see an operating system as opposed to a description existing only on paper. Second, an institution chosen for a visit needed to be involved in computing activities relevant to community colleges. Conceivably, the system could be applied at Delta. Three visitation sites received major attention, each meeting these criteria: Golden West Community College in the Los Angeles area, Dartmouth College in Hanover, New Hampshire, and the University of Illinois in Urbana, Illinois. Other sites visited, usually in connection with conferences held there, included: Cuyahoga Community College, Cleveland; University of North Dakota, Grand Forks; three Chicago City Colleges (Malcolm-X, Wright and Kennedy-King); and the University of California at Irvine.

In each visitation, the specific purposes varied. Golden West was viewed primarily as an example of a community college actively involved with instructional uses of computers. Naturally, the Team was interested in seeing what they were doing, what their experiences were, and what success they were having with instructional computing. While in the area, the Team made a visit to the University of California at Irvine to talk with Alfred Bork who employs the computer in teaching physics.

The visits to Dartmouth College and the PLATO project at the University of Illinois had a somewhat different purpose. While the Team was interested in experiences, successes and failures as at Golden West, attention was also devoted to exploring the possibilities of either connecting to or importing these systems at Delta. In connection with the PLATO project, the related visits to three campuses of the Chicago City College system were important since a number of PLATO terminals are being installed to make use of the central system in Urbana. Extensive plans are underway to use the PLATO system with community college students there and at Parkland College in Urbana.

Implications for Delta College

Area employers interviewing Delta students ask about experience with computers. Delta has a responsibility to its students to provide the opportunity for each student to say: "Yes, and it did these things for me," relating specific advantages of the computer as a tool for learning and for professional work. Delta College has a strong commitment to its students as individuals coming from diverse backgrounds with a variety of educational and vocational goals. The ultimate test of expanded use of the computer is found in the answers to the question: "What are the advantages to the student?" After extensive observation at many other sites, CITALA staff put forth a number of advantages to the student exposed to instructional computing.

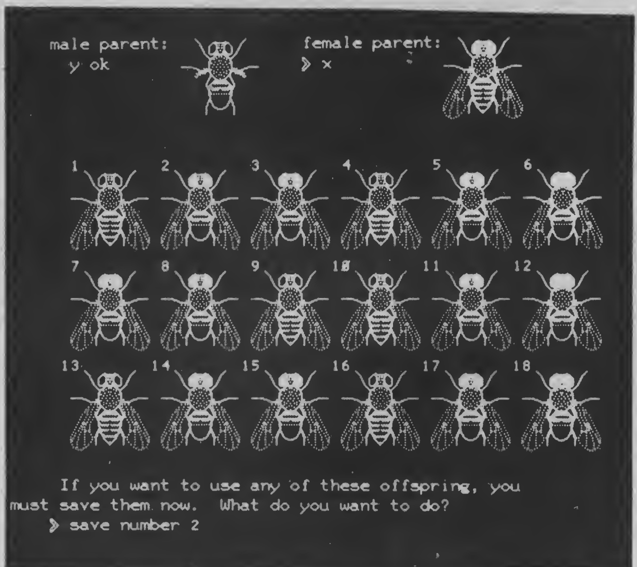
Greater individualization of instruction can be achieved. Each student has a program tailor-made for his own needs. The use of self-paced learning can be increased since each student proceeds through course materials and individual lessons as rapidly or slowly as he feels is appropriate. More individual student remediation is available, particularly where drill and practice are needed. In additional to the obvious advantages of individualized attention, students are exposed to simulations and problem-solving situations. Simulations enable students to interact with concepts and situations that otherwise must be dealt with as abstractions rather than as real-world phenomenon. As a high-speed calculating machine, the computer facilitates student problem solving, permitting direct interaction with problems of a type and size encountered in real life situations. By no longer being limited to exercise problems of pencil and paper solution, the student and instructor can use the computer as an aid in solving a wider range of problems. Students can manipulate large amounts of data rapidly, thus being able to experiment with these data in ways formerly not available.

Students can fail without embarrassment; as a consequence students who have particular learning difficulties can interact with a computer at a terminal, while they would not be willing to admit a learning problem to an instructor or counselor.

However, such benefits for the student are impossible without quality instructional materials and a computer system with certain physical characteristics critical to success:

a. terminals must be easily accessible for both student and faculty use;
b. terminals should be easy to operate, with clearly-labeled keyboards and instructions for use;
c. lessons and programs should be easy to understand;
d. the system should have a library of instructional programs available for student and faculty use;
e. the system should have an easily-learned, user-oriented language for authors of programs;
f. the system should lend itself well to both problem solving and computer-assisted instruction uses;
g. the system needs, in addition to computing power and text processing, the ability to display graphs or drawings and project other visuals (slide and/or microfiche).

The full report discusses alternative means of providing these capabilities: the MERIT Computer Network (or one of its host computing centers); the Dartmouth Time Sharing System; the PLATO IV System; a commercial time-sharing service with special educational rates; and an expanded IBM 360/40 (Delta's present system). Other possibilities were considered but with less detail: a stand alone minicomputer; a different on-site computer system; a split system with one medium size computer for administration and another for instruction.



Biology via PLATO terminal. The characteristics of offspring are shown in a fruit fly genetics experiment.

Expanding Instructional Uses

The Project Team realized that students, faculty, and staff of Delta College needed experience with the computer as an instructional tool. Thus, conclusions at the close of the investigation were quite positive. The Team recommended implementing a three-year, pilot project designed to expand instructional computing at Delta. The final report spelled out a plan that included immediately leasing four ports on the Dartmouth Time Sharing System, leasing six terminals for use on the Dartmouth System or elsewhere, and ordering four PLATO IV terminals. Under the supervision of a Project Director, the Delta staff would develop and implement in-service training programs and incentives to encourage faculty involvement in using the computer as a teaching tool. The final stages of the pilot project would concentrate on evaluative measures.

The rationale for the recommendations may be of more interest than the details. Students, faculty, and staff of Delta College need experience with the computer as an instructional tool. The best way to acquire this experience seems to be to lease time on an educational time sharing system with extensive educational software available and appropriate for college use. Among the various alternatives, only the Dartmouth and PLATO systems have extensive software for community colleges; the Dartmouth System was immediately available.

The crucial questions regarding remote computing relate to timing. Today university computing centers provide rich resources for a wide range of users; however, they tend to charge more for the most straightforward kinds of instructional computing than a small computer dedicated to the purpose. Eventually the rates charged by university computing centers will come down, especially as the universities provide special computing systems or means of access for just these limited kinds of computing on their own campuses.

The capabilities of small computers are increasing while costs continue to go down. Today one can get a "personal" computer for under $4,000 but not very much interesting software is available with it and storage of large files of data and programs is inconvenient. In a few years the cost of such a personal computer will drop to below $1,000 and the capabilities will increase markedly at least in special areas of use for which vendors anticipate large markets. These individual computers and small clusters of terminals on slightly larger machines will provide an important part of instructional computing on college campuses.

A prediction about the mixture of networks and mini-computers in the next ten years is further complicated by difficulties in predicting communication costs. Service networks designed for communications between users and their computers, and perhaps non-profit organizations serving educational institutions, will make reliable and low-cost communications available just for the purpose of access to networks. Until this happens, a college like Delta is rather "distant" in communication dollars from the serviceable networks.

Since the relations among cost, software, storage and communications will shift in the next five to ten years, the future is difficult to judge. Delta would like to develop multiple sources of computing: a college facility for moderate-sized problems; occasional minicomputers for departments, special courses, or other situations in which some cost-saving can be achieved; and telephone access to large computers for problems requiring special software systems or a large data base. At the same time, programs and data should be able to migrate from one system to another, so that the specialized data base at a university can be moved to the local computer when execution locally will be more economical. The software developed on the local computer for instruction might later be programmed into the minicomputer for more economical execution with students in a given unit of the college.

The sharing of materials, programs, and ideas with other authors, programmers, and users will be of considerable value. Because both the Dartmouth and PLATO systems serve networks of users similar to Delta College, the opportunities for sharing with other users in the network are much greater than for "stand-alone" and commercial time-sharing systems.

In short, the rationale behind Team recommendations relies on the assumption that instructional uses of computing are evolving rapidly and Delta will benefit from experimenting with various innovations in equipment, software and applications.

The participants in Project CITALA were Doug Anderson, M. Gene Arnold, Mark Baldwin, Darrell Berry, Bruce Corliss, Robert DeVinney (Chairman), Gayle Hanna, David Howard, John Kostoff, Craig McClain, Dennis McNeal, and Betsy Smith.

Decision-Making

Nay, lad! *Deciding's* not your ploy,
For that's a risky game.
It's *making a decision*
That's your surest road to fame.

Decide means *to take action,*
And that might rock the boat,
And you act and don't succeed,
Small chance you'll stay afloat.

But . . . *making a decision.*
Ah! *That's* the way to swing.
It keeps the masses happy
And doesn't change a thing.

So get yourself a task force
Well-skilled in all the arts
And call them all together
And watch them flip their charts.

For Jack says no and Jim says yes
And Billy says perhaps
And Chester asks good questions
. . . When he isn't taking naps.

And Bertram, chomping his cigar,
Is chock full of statistics,
While Waldemar, who smokes a pipe,
Is famed for his heuristics.

"The figures prove. . . ," "The model says . . ."
"The forecast bears me out."
"The complex simplex program
Shows I'm right without a doubt."

Let's tiptoe out and close the door
And let them stew a while.
No fear that they'll do something rash,
For *doing's* not their style.

Reality's an untamed beast
That's difficult to master,
But models are quite docile
And give you answers faster.

So build yourself a model
To glorify your name.
Then get yourself a task force
And learn to play the game.

—J. C. L. Guest

Another New Game From CREATIVE COMPUTING ....

BINARY

by Ted C. Park
Pacific Union College
Angwin, California

DESCRIPTION

This game tests your skills in binary-to-decimal and decimal-to binary conversion. You will be given twenty conversion trials. Numbers are chosen randomly and your score will be printed at the end. The answer to any conversion you miss will be displayed; if the next conversion is presented, you may assume you got the previous one correct.

SUGGESTED MODIFICATIONS

1. If your computer has an "ENTER" statement or some other sort of timed input, then fix-up this program to allow only a certain time for each conversion and then go on to the next one.
2. Allow the user to enter the range (number of bits) allowed for the binary numbers.
3. Being able to select the number of conversions of each type would be beneficial.
4. Extend this exercise to other bases!
5. Modify program to check for duplicate numbers.

LISTING

```
BINARY

100  DIM B$[2],B[5],I$[72]
110  B$="01"
120  T0=20
130  PRINT
140  PRINT
150  FOR I=1 TO 10
160  GOSUB 560
170  PRINT "BINARY:     ";
180  FOR J=1 TO 5
190  PRINT B$[B[J]+1,B[J]+1];
200  NEXT J
210  PRINT "      DECIMAL:    ";
220  INPUT A
230  IF A=D THEN 260
240  PRINT D
250  T0=T0-1
260  PRINT
270  NEXT I
280  PRINT
290  PRINT
300  FOR I=1 TO 10
310  GOSUB 560
320  PRINT "DECIMAL:  ";D;
330  PRINT "      BINARY:     ";
340  I$="00000"
350  INPUT I$[6]
360  IF LEN(I$)>10 THEN 420
370  I$[11-LEN(I$)]=I$[6]
380  FOR J=1 TO 5
390  IF B$[B[J]+1,B[J]+1]#I$[J,J] THEN 420
400  NEXT J
410  GOTO 480
420  PRINT " ";
430  FOR J=1 TO 5
440  PRINT B$[B[J]+1,B[J]+1];
450  NEXT J
460  PRINT
470  T0=T0-1
480  PRINT
490  NEXT I
500  PRINT
510  PRINT
520  PRINT "YOUR SCORE:";INT(T0/.2+.5);"%"
530  PRINT
540  PRINT
550  STOP
560  D=0
570  FOR J=1 TO 5
580  B[J]=INT(RND(0)+.5)
590  D=D*2+B[J]
600  NEXT J
610  RETURN
620  END
```

SAMPLE RUNS

```
RUN
BINARY


BINARY:     00111      DECIMAL:    ?14
 7

BINARY:     11100      DECIMAL:    ?26
 28

BINARY:     11000      DECIMAL:    ?24

BINARY:     10011      DECIMAL:    ?19

BINARY:     11100      DECIMAL:    ?28

BINARY:     01101      DECIMAL:    ?13

BINARY:     11010      DECIMAL:    ?26

BINARY:     10010      DECIMAL:    ?18

BINARY:     10000      DECIMAL:    ?16

BINARY:     10110      DECIMAL:    ?22


DECIMAL:  25       BINARY:     ?11001

DECIMAL:  3        BINARY:     ?11

DECIMAL:  7        BINARY:     ?111

DECIMAL:  8        BINARY:     ?1000

DECIMAL:  3        BINARY:     ?11

DECIMAL:  21       BINARY:     ?10101

DECIMAL:  15       BINARY:     ?1110
 01111

DECIMAL:  4        BINARY:     ?100

DECIMAL:  8        BINARY:     ?1000

DECIMAL:  12       BINARY:     ?1100


YOUR SCORE: 85   %


DONE
```

Shorthand Instruction via Computer

by
C. Bruce Kavan & Leona M. Gallion
Indiana State University

Introduction

A unique application of the computer to the classroom environment has been to utilize a computer-based system for writing instructional materials for beginning shorthand classes. This system is used to verify and statistically analyze instructional new-matter shorthand dictation materials which are vocabulary controlled. These materials are then used for instructional purposes in beginning shorthand classes.

Research Basis

Research[1] has established that achievement in beginning shorthand increases with the use of vocabulary-controlled dictation materials. However, only a limited amount of this type of material is currently available. This may, in part, be due to the large expenditure in human capital necessary to construct, verify, and analyze dictation materials utilizing a limited or controlled vocabulary. To facilitate and encourage the preparation of vocabulary-controlled dictation materials, a computer-based system was designed and implemented at Indiana State University in the Fall of 1973. This system has been used subsequently in shorthand methods classes and workshops for writing new-matter vocabulary-controlled dictation materials.

The Computer-Based System

The computer-based system is built upon a word base dictionary composed of the 1500 most frequently used words in written business office communications as established by Mellinger.[2] The other information in the word base dictionary was researched and compiled by the designers from the various shorthand publications. The dictionary for these selected words contained

a. the word image
b. the lesson number in which the word could first be written in beginning shorthand
c. the number of syllables in the word
d. the word frequency grouping in hundreds
e. the word type(s) — brief form, brief form derivatives, word endings, and word beginnings

Design criteria for the system was based on the following two objectives:

1. To design the necessary computer software which comprises the components of the system. Each of these components or subsystems consists of one or more computer phases or programs. The objectives of the subsystem were
   a. to statistically analyze dictation material.
   b. to access the dictionary (see Figure 1 for sample page of the dictionary)
   c. to provide the working tools for writing dictation
2. To design a computer system with maximum simplicity of operation for use by the novice student user while simultaneously achieving maximum efficiency of computer resources.

The system is used first to verify a passage of dictation material which has been coded for use in a specific lesson in beginning shorthand. If a word in the dictation passage is not in the dictionary or is a word that cannot be written either in or prior to the lesson introduction code, that word will be underscored by astericks in the output (see Figure 2).

After all words in a dictation piece are only those among the 1500 most frequently used words and are those which can be written in either the introductory lesson or the previous lessons, the statistical phase will execute. The edited text image is then outputted into standard word groupings of 10 for constant-level dictation (indicated by /) and 20 for traditional dictation (indicated by /#/). (see Figure 3) The syllabic intensity of the passage is computed as well as the following statistics: percent of words from each lesson, percent of words in each hundred of the 1500 most frequently used words as well as the percent of brief forms, brief form derivatives, word endings, and word beginnings. Further, all words that can first be written in each of the lessons are listed as well as brief forms, brief form derivatives, word beginnings, and word endings (see Figures 4 to 8).

References

1. Leona M. Gallion and Alberta Anderson, "Controlled Vocabulary Beginning Shorthand Dictation," *Journal of Business Education,* October, 1972, pp. 27-28.
2. Morris Mellinger, *Basic Vocabulary for Written Business Office Communications,* (Chicago: Chicago State College Publication Series, 1970.

WORD	LESSON WORD FIRST USED IN VOL. I, D.J.	NUMBER OF SYLLABLES IN WORD	WORD FREQUENCY (100*S)	WORD TYPE
A	3	1	1	1
ABLE	3	2	3	
ABOUT	15	2	1	1
ABOVE	13	2	2	
ACCEPT	5	2	5	
ACCEPTANCE	21	3	14	
ACCEPTED	14	3	10	
ACCIDENT	21	3	13	
ACCIDENTS	21	3	14	
ACCOMMODATE	20	4	14	
ACCOMMODATIONS	27	5	14	3
ACCOMPLISHED	20	3	14	
ACCORDANCE	11	3	10	
ACCORDING	11	3	4	3
ACCORDINGLY	37	4	12	3
ACCOUNT	20	2	2	
ACCOUNTING	20	3	9	3
ACCOUNTS	20	2	9	
ACKNOWLEDGE	23	3	9	1
ACROSS	5	2	10	
ACT	5	1	11	
ACTION	9	2	4	3
ACTIVE	5	2	15	
ACTIVITIES	5	4	6	
ACTIVITY	5	4	11	
ACTUAL	31	3	8	3
ACTUALLY	31	4	9	3
ADD	5	1	7	
ADDED	14	2	8	
ADDITION	27	3	3	3
ADDITIONAL	27	4	2	3
ADDRESS	5	2	4	
ADDRESSED	5	2	11	
ADEQUATE	14	3	12	
ADJUSTMENT	19	3	10	3
ADMINISTRATION	16	5	7	3
ADMINISTRATIVE	16	5	13	
ADVANCE	5	2	8	
ADVANTAGE	21	3	6	1
ADVANTAGES	21	4	15	2

Figure 1. Sample page of the dictionary.

```
         E D I T E D                   T E X T                   I M A G E

DEAR SIR:
     ****

YOUR LETTER OF MAY FIRST CAME TO ME THIS DAY . IT WILL BE / ONLY THREE
                         ****

DAYS BEFORE I CAN MAIL YOU THE TWO HUNDRED DAILY /01/ PAPERS FOR WHICH YOU
                                   *******

ASKED . IS THIS ALL RIGHT WITH YOU ? I WOULD / PUT A FAST SERVICE ON
                                                     ****

THIS PACKAGE . THE COST WILL BE THREE DOLLARS /02/ FOR THE FIRST ONE
                                                                 ***

HUNDRED , BUT ONLY TWO DOLLARS FOR THE REMAINING / ONE HUNDRED .
*******                                            *** *******

CORDIALLY ,


NUMBER OF TEXT CARDS READ ------------------------     7
NUMBER OF GOOD WORDS ON TEXT CARDS ---------------    60
NUMBER OF WORDS REJECTED ON TEXT CARDS -----------     8
NUMBER OF GOOD SYLLABLES ON TEXT CARDS -----------    74
SYLLABIC INTENSITY (GOOD SYLLABLES / GOOD WORDS) -    1.233
```

Figure 2. Computer output of edited text image showing rejected words.

```
         E D I T E D                   T E X T                   I M A G E

DEAR SIR :

I READ YOUR LETTER OF MAY FIRST THIS DAY . IT WILL / BE ONLY THREE

DAYS BEFORE I CAN MAIL YOU THE TWO DAILY /01/ PAPERS FOR WHICH YOU

ASKED . IS THIS ALL RIGHT WITH YOU ? I / WOULD PUT GOOD SERVICE ON

THIS PACKAGE . THE COST WILL BE THREE /02/ DOLLARS FOR THE FIRST ,

BUT ONLY TWO DOLLARS FOR THE / REMAINING .

CORDIALLY ,


NUMBER OF TEXT CARDS READ ------------------------     7
NUMBER OF GOOD WORDS ON TEXT CARDS ---------------    61
NUMBER OF WORDS REJECTED ON TEXT CARDS -----------     0
NUMBER OF GOOD SYLLABLES ON TEXT CARDS -----------    75
SYLLABIC INTENSITY (GOOD SYLLABLES / GOOD WORDS) -    1.229
```

Figure 3. Computer output of edited text image showing usable passage.

GROUP	FREQUENCY	PERCENT FREQUENCY
1	41	67.2
2	4	6.5
3	5	8.1
4	1	1.6
5	2	3.2
6	1	1.6
7	1	1.6
8	1	1.6
9	1	1.6
10	0	0.0
11	0	0.0
12	1	1.6
13	3	4.9
14	0	0.0
15	0	0.0
TOTAL	61	

Figure 4. Computer output of word frequency count by hundred groupings.

LESSON	FREQUENCY	PERCENT FREQUENCY
8	17	27.8
7	5	8.1
5	22	36.0
4	2	3.2
3	9	14.7
2	4	6.5
1	2	3.2
TOTAL	61	

Figure 5. Computer output of lesson frequency count.

```
LESSON  08          LESSON  05          LESSON  03

  BE                  ASKED               DAYS
  BEFORE              BUT                 DEAR
  CORDIALLY           CAN                 I
  DAILY               FIRST               IT
  FOR                 IS                  PAPERS
  GOOD                LETTER              WILL
  ONLY                OF
  PUT                 PACKAGE
  THIS                SERVICE           LESSON  02
  WHICH               SIR
  WOULD               THE                 MAIL
                      THREE               READ
                      WITH                REMAINING
LESSON  07            YOU                 RIGHT
                      YOUR
  ALL
  COST                                  LESSON  01
  DOLLARS           LESSON  04
  ON                                      DAY
                      TWO                 MAY
```

Figure 6. Computer output of word usage by lesson.

```
TYPE 1                THIS              TYPE 3
                      WHICH
  BE                  WILL                CORDIALLY
  BUT                 WITH                DAILY
  CAN                 WOULD               ONLY
  FOR                 YOU
  GOOD                YOUR
  I                                     TYPE 4
  IS
  IT                                      REMAINING
  OF                TYPE 2
  PUT
  THE                 BEFORE
```

Figure 7. Computer output of word forms by type.

TYPE	FREQUENCY	PERCENT FREQUENCY
1	31	83.7
2	1	2.7
3	4	10.8
4	1	2.7

Figure 8. Computer output of word forms by count.

Profile of an Industry

Though its origins go back several centuries, the digital computer is primarily a product of technological innovation during the last two decades. Its use has grown exponentially because the information problems it helps to solve have grown that way. Few people, twenty years ago, could foresee either the need for such a powerful problem-solving tool or that the computer could satisfy the need.

In 1950, only a handful of computers were in use helping scientists and mathematicians speed routine calculations. Some people thought that was all the machine was good for. But in 1951, the Bureau of the Census received its first electronic digital computer—a UNIVAC I. Computer usage, related services and the number of companies that provide computers and supporting services have been multiplying ever since.

It is now estimated that well over 100,000 computers are in use worldwide.

How did it happen so fast? Why so many computers so soon?

For thousands of organizations, for hundreds of thousands and perhaps even millions of individuals, the problem has been the management of information. Information has proliferated. And we all need help to sort out information, store it, process it, analyze it and locate it fast.

Above, computer operator is a career opportunity created by the computer industry. This is a good place for high school graduates to enter the field.

Below and facing page, computer technology continues to improve, year by year. Internal operating speeds of large systems are now measured in billionths of a second.

Today, wherever you look, there's an information problem and a need for swift, efficient information handling—whether you're trying to capture, store and analyze many hundreds of thousands of bits of information about a jet engine test . . . to make sure that the several thousand items that customers in a food market want and need are there when they want and need them . . . to analyze and report data collected by pollution control instruments . . . to calculate the best design for a bridge, a building or a bulldozer . . . to catalog, index and quickly find facts in a library . . . to handle the paperwork involved in running a government agency . . . to relieve doctors and nurses of time-consuming clerical work . . . to help farmers to breed animals and raise crops with greater efficiency and productivity . . . and on and on, endlessly.

Many people with problems like these find the large storage capacity, the logic power and the electronic calculating speed of today's computers useful in handling growing masses of data quickly and efficiently, to get needed information in time to take meaningful action, to solve problems before they become crises.

Computer Improvements

For many years the uses of computers have expanded in both quantity and variety. Also, there has been a rapid and steady flow of improvements in computer technology and organization. Computer internal operating speeds are now measured in billionths of a second—just ten years ago, millionths of a second was considered fast.

Far more computer power is now packed into far less space. Internal circuits now must be examined with microscopes in the faster computers where many electronic circuits can be packed on a single chip of silicon little more than a tenth of an inch square.

Main computer memory can now store millions of characters of information. There is quick access to hundreds of millions of additional characters in disk storage units.

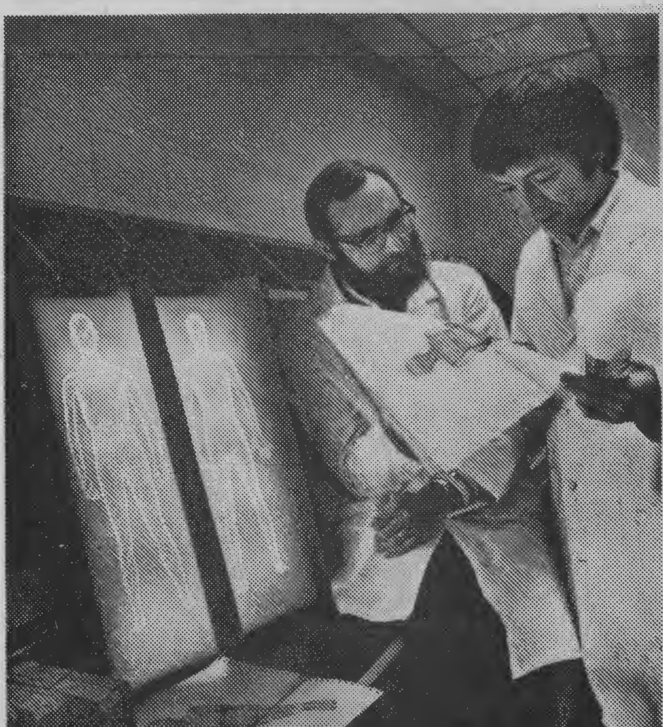
Industry Growth

In 1950 there were only a handful of companies in the computer business. Today there are many hundreds. They develop and build computing systems, prepare programs to instruct the machines, operate service bureaus, and sell peripheral equipment such as tape units, disk storage units, printers, and display terminals. Other firms provide consumable supplies, such as punched cards and magnetic tape, and engage in other aspects of the industry.

Many thousands of companies, of course, supply products and services to manufacturers and users of computers. Makers of electronic components are an obvious example. Also needed are paper, fabricated metal impossible alternative, the instructor may give him a command that makes no systematic contributions to subsequent data values.

The Pedagogy: a Simulated Scientific Community

The classroom structure of the course is that of a scientific community where each student plays the role of an individual researcher. The student participates in activities designed to parallel those of a scientist in a real community. For example, the student is directed toward a particular problem area. He becomes familiar with at least some of the literature in that area and examines current theories and research bearing on it. He is acquainted with some of the costs associated with conducting research in that field, and he is informed of the resources available to him. These resources, usually in the form of points, are designed to parallel those known to exist in the real world. He proposes research that he would like to conduct, bringing to bear what he is learning about the subject matter, research design principles and his available resources. He designs a research program, not merely isolated experiments. He argues for his research strategy articulating how he believes such a strategy will accomplish his stated research goals. Upon receiving his "contract", he proceeds to implement his strategy by conducting the experiments he has proposed. He updates his knowledge about the area by re-examining or modifying later experiments, based on what he has learned from his own research and other "scientists" in his "community". He communicates his updated knowledge to the rest of the "community" in the form of research reports written in a format acceptable for publication, through formal presentation at "conventions" and through informal bull sessions with other members of the community. In this communication, he not only articulates his research goal, hypotheses tested, and experimental designs, but his method of data analysis and his conclusions. On the basis of updated knowledge, he plans his next experiments, taking into account the costs of such experiments, and the cycle is repeated. One of his final scientific communications may be a review paper that summarizes the state of knowledge of his scientific community in a particular area. It may be a report to a sponsor. Within a two-month period he will have designed and analyzed the results of about ten experiments that he himself has conducted on several problem areas.

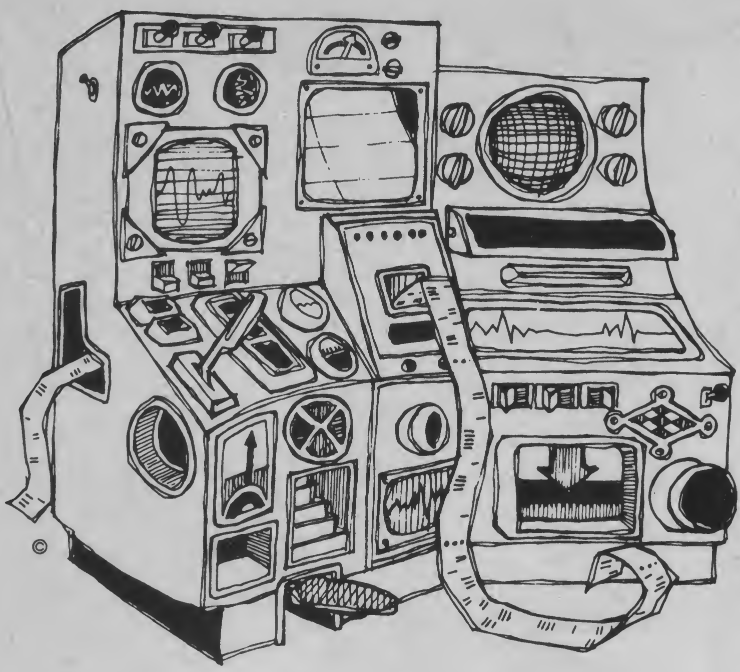


COMPUTERS IN MEDICAL RESEARCH
Scientists at the Texas Institute for Rehabilitation and Research in Houston are using computer produced three-dimensional measurements to study medical applications ranging from deformities in children to weight loss in astronauts. Dr. R. E. Herron, left, Director of the Institute's Biostereometrics Laboratory and Professor Jaime R. Cuzzi, study contour maps of the body which can be produced in other ways such as a graph showing how body volume is distributed from head to foot and cross sections or slices of the body. (Photo IBM)

The simulated scientific community is facilitated by the use of computer data-generating models. The student rehearses all of the major roles of the scientist except the very time-consuming and expensive data collection step. (Even this step can be included in some problems and in some sessions of the course, but the simulation does not depend on its inclusion.) This process is summarized below:

Sequence of Student Activities in Simulated Scientific Community

1. Orientation to the Scientific Community
   a) Activities involved in participating in the scientific community.
   b) Familiarity and experience with the computer in terms of running EXPER SIM experiments.
   c) Discussion of issues and problems associated with the particular model in the library that will be used.
   d) Explanation of simulated costs and budget.
2. Students submit a proposal of their research plan for approval by the instructor.
3. Upon approval of research plan, students submit their first design to the computer and write a short report that will either be posted or dittoed for all class members. If the latter, then it may be viewed as Vol. 1 of a journal for which the students can select a name and formulate editorial policy.
4. Students, now working individually and in collaboration with other students, submit the next design to the computer. It should be based on what has been learned from the first set of experiments. This report goes into Vol. 2.



5. Students design a third experiment based on knowledge gained previously and within the resources of their budget. The report on this design goes into Vol. 3.
6. Students review the "literature" generated in their class and write a review paper. The paper not only discusses their own research, but that conducted by other classmates.
7. An optional exercise is to have students write a report to their sponsor justifying their own research and the expenditure for that information. This may be reserved for students who have overspent their "budgets" or who have conducted a series of disjointed or trivial experiments which the instructor believes needs to be rationalized.

Building New Models

Both MESS and LESS have been written so that new models can be added to the library by writing FORTRAN subroutines to operate with the supervisor or management program. In other words, a professor or advanced student can write his own model as a FORTRAN subroutine for the supervisor program. He need not concern himself with the programming required to interface the student with the data-generating model.

For instructors or students who do not program or do not wish to commit themselves to a programming task, it is possible to develop and build new models for the library making use of the Simulation Writer Interactive Program (SWIP) written by Robert Stout and developed at the University of Michigan. For smaller computers, LESS has features that would enable the building of certain kinds of models. Both are possible in the interactive mode.

The development of SWIP and the LESS programs makes it possible to add models to the library that are concerned with physics, chemistry, biology, medical sciences, and engineering as well as psychology, sociology, economics, education, and political science.

Such models can be used in methodology and statistics courses as well as specific content courses where it is desirable to teach students by way of a method of inquiry.

Costs

The cost of using MESS and LESS once they are implemented is very small compared to the cost of maintaining a traditional laboratory course.

The programs are designed for batch processing (requiring the use of a keypunch or equivalent) as well as interactive processing (requiring the use of a teletype or other user terminal). At the University of Michigan both methods are used simultaneously in the psychology course. (A sample batch run follows this article.) The parameters of an experiment can be conveyed to the computer on about a dozen cards, many of which can be used over and over again. In the interactive mode the student is prompted for the input required, and error messages are immediately available. The cost of batch processing student experiments is usually many times cheaper than the interactive mode. Its disadvantages are a longer turn-around time and delayed access to error messages. Although the interactive mode is more expensive, the student obtains "results" from his experiment more quickly. Even with this more expensive mode, a budget of $7.00 to $10.00 per student has been sufficient for a given term. Cost figures on SWIP will be available soon.

The programs can be obtained for a nominal fee. For information on MESS and SWIP write: Dr. Dana B. Main, 3435 Mason Hall, Department of Psychology, University of Michigan, Ann Arbor, Michigan 48104. For information on LESS write: Dr. Arthur Cromer, Computing Center, University of Louisville, Louisville, Kentucky 40208.

Acknowledgement

EXPER SIM was developed in conjunction with the center for Research on Learning and Teaching at the University of Michigan with support from the Exxon Education Foundation.

[Ed. note: the IMPACT Program of the Exxon Education Foundation has selected EXPER SIM as a tried and effective innovation for which adoptions are encouraged. IMPACT provides modest grants to potential adopters of EXPER SIM (and other items on their adoption "menu") on the basis of proposals received. Contact Ms Caryn Korshin at the Exxon Education Foundation for further details.]

(A sample run of EXPER SIM is shown on Page 46.)

EXPER SIM Sample Run

```
MICHIGAN EXPERIMENTAL SIMULATION SUPERVISOR
      VERSION 3-SB   JANUARY, 1973
```

These lines give program title and authors' credits.

```
      IMPRINTING SIMULATION
      D. W. RAJECKI, FALL, 1970
(d)   MODIFIED BY BOB STOUT, AUGUST, 1972

 ENTER SUPERVISOR COMMAND
>>EXPT
```

Here the program asks for instructions. Card (d) is printed, which tells the program to begin an experiment.

```
(e)

  ENTER EXPERIMENT TO LINE
  A. EINSTEIN SECT. 029 TARG-ARO EXPT
```

The program asks for an identification line and card (e) is printed.

```
    ENTER NO. OF EXPERIMENTAL CONDITIONS
(f) 4
```

This experiment is to have four conditions, as indicated on card (f).

```
    DEFINE EXPERIMENTAL CONDITION(S)
    TARG=CYL, HEN  ARO=3,5 WALK=mat
(g) @END
       4 CONDITION(S) DEFINED
(h)
```

The program asks for experimental condition definitions. Cards (g) and (h) are printed.

The program agrees that four conditions have been defined.

```
   THE FOLLOWING VARIABLE SETTINGS ARE CONSTANT ACROSS ALL CONDITIONS:
        REARING=SOCIAL        INDUCT=MECH           WALK=MATCHED
            AGE=RANDOM
           TEST=1.000

     VARIABLE SETTINGS FOR CONDITION  A
         TARGET=CYLINDER
        AROUSAL= 3.000

     VARIABLE SETTINGS FOR CONDITION B
         TARGET=HEN
         AROUSAL= 3.000

     VARIABLE SETTINGS FOR CONDITION C
         TARGET=CYLINDER
        AROUSAL=5.000

     VARIABLE SETTINGS FOR CONDITION D
         TARGET=HEN
        AROUSAL= 5.000
```

The program states which variables are to remain constant across all the four conditions. Note that the variables--REARING, INDUCT, and AGE--have been set to their default values.

Each condition is defined. Note that the variable TARGET alternates first between CYLINDER and HEN, then the variable AROUSAL alternates between 3 and 5.

```
(i)  ENTER NO. OF SUBJECTS IN EACH GROUP
     15
```

The programs asks for the number of subjects to be run in each condition, and then prints card (i).

```
   A. EINSTEIN SECT. 029 TARG-ARO EXPT
   16:59.08  APR 23, 1973
   GROUP NUMBER  1
   CONDITION(S):  A
   NUMBER OF SUBJECTS:   15
```

The experimental simulation begins: the first line is the identification line, then comes the time and date of the experiment. Group Number 1 is to be run under condition A and has 15 subjects.

```
      TEST1   SCORES
           2.00     2.80     0.600     1.30    1.30
          0.500    0.700     3.70      1.70    2.50
           1.10    0.900     1.40      6.40    5.20
```

The score for each subject is printed out.

```
      NO. OF SS WITH COMPLETE DATA:  15

      VARIABLE:  TEST1
      MEAN:  2.140
      VARIANCE:  3.031
      STD. DEVIATION:  1.741
```

The scores are analyzed and the statistics are printed out.

```
   A. EINSTEJN SECT. 029 TARG-ARO EXPT
   16:59.09  APR 23, 1973
   GROUP NUMBER  2
   CONDITION(S):  B
   NUMBER OF SUBJECTS:  15

      TEST1   SCORES
            4.30     1.50    3.30    1.50      3.10
           0.700     1.90    4.00    4.00     0.200
            5.10     0.0    0.600    1.50      2.60

      NO. OF SS WITH COMPLETE DATA:  15

      VARIABLE:  TEST1
      MEAN:  2.287
      VARIANCE:  2.613
      STD. DEVIATION:  1.616

   A. EINSTEIN SECT. 029 TARG-ARO EXPT
   16:59.09  APR 23, 1973
   GROUP NUMBER  3
   CONDITION(S):  C
   NUMBER OF SUBJECTS:  15

      TEST1    SCORES
             6.90     12.6     7.60     1.20     9.00
             10.7     7.50     1.20     7.50     9.50
             6.50     1.90    0.900     1.30     10.3

      NO. OF SS WITH COMPLETE DATA:  15

      VARIABLE:  TEST1
      MEAN:  6.307
      VARIANCE:  15.93
      STD. DEVIATION:  3.992

   A. EINSTEIN SECT. 029 TARG-ARO EXPT
   16:59.09  APR 23, 1973
   GROUP NUMBER  4
   CONDITION(S):  D
   NUMBER OF SUBJECTS:  15

      TEST1    SCORES
             7.20     8.40     7.40    0.800     8.10
             9.90     12.7     7.50     1.10    0.600
             8.40     0.0      12.0     0.0      7.50

      NO. OF SS WITH COMPLETE DATA:  15

      VARIABLE:  TEST1
      MEAN:  6.107
      VARIANCE:  19.43
      STD. DEVIATION:  4.408
```

The output for conditions B-D follows the same format as the output for condition A.

```
   EXPERIMENT COMPLETED.
```

Notification of completion of the experiment. The user knows that all went well with this run.

```
(j) ENTER SUPERVISOR COMMAND
    >>STOP

       NUMBER OF EXPERIMENTAL RUNS     1
       NUMBER OF GROUPS SIMULATED      4
```

The program asks for another simulation supervisor command. Card (j) is printed and the program tallies the number of experiments and groups simulated.

```
    $SOURCE PREVIOUS
(k) $SIGNOFF
```

The user signs off with card (k) and MTS prints out the signoff statistics.

```
**** ON AT 16:59.48      04-23-73
**** OFF AT 16:59.18     04-23-73
**** ELAPSED TIME           .483 MIN.
**** CPU TIME USED         4.441 SEC.         $.35
**** CPU STOR VMI          1.816 PAGE-MIN.    $.09
**** WAIT STOR VMI          .162 PAGE-HR.
**** CARDS READ           20                  $.02
**** LINES PRINTED       227                  $.10
**** PAGES PRINTED        11                  $.04
**** DRUM READS          114
**** APPROX. COST OF THIS RUN IS     $.58
```

Still a Few Bugs in the System

It bugs us here at Creative Computing when magazines, newspapers, and commentators blame various mistakes on the computer. Nor are members of the mass media the only ones who point the accusing finger at the computer — people in government, industry, and schools are guilty as well. The computer is a convenient scapegoat when the error actually may lie with the programmer, keypuncher, faulty data collection techniques, or any one of a dozen points only one of which is the computer itself.

In this column, we'll reprint articles or quotes which blame various catastrophies on the computer. It's up to you, the reader, to decide whether the computer is actually to blame. (Also, if you spot an appropriate article for the "BUGS" column, please send it in.)

A parting note: in *your* conversation and writing, please be sure to differentiate between computers, programs, people, and data.

Our BUGS quotes this month are taken from the July 1974 issue of *Road & Track* and a recent article in the *South Bend Tribune*. What do you think — were they computer errors?

* * *

In Swansea, Wales, Barry Carr was quite happy when he passed his driving test soon after his 17th birthday, the earliest age at which Britons are permitted to drive a car. But when his license arrived, it bore 12 endorsements for a whole array of driving **OFFENSES**, plus a 28-day driving suspension. Police proved sympathetic when it was found a computer at the license office had run wild. "The system has not been operating for long," said an official. *Road & Track.*

* * *

Fred Finn Mazanek, a one-year-old guppy, died, recently, leaving an estate of $5,000.

Stan Mazanek, twenty-four, a student at the University of Arizona, had filled out an insurance form he received in his mail box marked "Occupant," entering the fish as the insured party. No fraud was involved in the policy. The guppy's age was listed as six months, his weight as thirty centigrams, and his height as three centimeters.

The Globe Life and Accident Insurance Co. apparently issued Policy No. 3261057 in Fred Finn's name through a computer error. When Mazanek filed a claim following the guppy's demise, they sent a sales representative to see him to find out if he was the sort of person who would take advantage of a clerical error.

He was. The company settled out of court for $650. *South Bend Tribune.*

Pocket Calculator TRICKS !!

1. Enter 107734, turn calculator 180 degrees and read.
2. Enter 4. Press the addition button. Enter 57734. Turn the calculator 180 degrees, read what it says. Press addition button again and read.
3. If the price for 28,430,938 barrles of oil is increased 2.5% who benefits? Turn calculator 180 degrees for answer.

Can you invent some new tricks? You'll have to settle for words containing only the letters B, E, H, I, L, O, and S. Send your results to *Creative Computing*.

We heard that Hewlett-Packard technicians play a game on the company's programmable HP65 pocket calculator in which two players subtract one to three objects each turn from a starting total of 15 in an effort to leave the loser with the last turn. If the player in the role of the calculator wins, the upside down reading says BLISS. If the other player wins, the reading is I LOSE.

Answers:
(1) HELLO (2) HELLS BELLS (3) SHELL OIL.

> **Wisdom is OK for places where you have to be wise, but it isn't so good where you have to know.**
>
> **Kenneth Boulding**

COMPUTERS IN ROAD BUILDING
Using a problem-oriented language such as COGO, civil engineers can efficiently determine the cut and fill, gradients and curvatures for a new road, pipeline, or transit system. (Photo Caterpillar)

A Universal Word Game in BASIC

by Barney M. Milstein
Associate Professor of Literature
Stockton State College
Pomona, N. J.

This article describes an interactive paedagogical game written in BASIC for vocabulary building work in any natural language. Although the game as presented here is intended for antonym-matching, it can easily be modified to test definitions, either in the target natural language, or in the native language, as with remedial work. The version discussed below is for German; the program is named SPIEL1.

The same is played by presenting the student with three rounds of ten words each. Lexical items are ordered in each of three groups according to increasing difficulty. In order to advance a round, the student must match at least seven correct answers on the first attempt; otherwise he is ejected from the game at the end of the respective round. In case of errors, the correct answer is displayed for the student, and he must enter it before proceeding to the next item. At the end of three rounds, the student is given a 'rating' on a scale from 15 to 30. The parameters of the ratings, as well as those for advancing a round, are arbitrary and alterable at the will of the programmer.

Items are selected at random from a data array. In order to keep items from repeating, a flag is set on both the question and its antonym (i.e. the correct answer).

The reader is asked to consult the accompanying program listing for references to specific line numbers. The program as it appears here is implemented on the E. I. S. network in New Jersey.

Lines 70-280 contain instructions, and are self-explanatory. They should be modified in appropriate places for use with other languages, or with variations.

If the parameters for the ratings (lines 280-350) are altered, accompanying changes must be made in the conditions set in lines 790-860: these determine which message will be displayed to the student at the conclusion of the game.

Lines 360-380 are used in conjunction with the generation of a random number. In the CALL-OS system as used by E. I. S., the 'rnd' function can be provided with a 'throwaway' variable ('a'); this will produce a different sequence of random numbers if a different number for 'a' i s entered each time the program is run.

The Dimension Statement is routine. In this case 216 items are present in the German array, with the variable 'w$' being the literal string. The variable 'f' i s used to flag items for non-repetition.

Lines 410-440 initialize the three counters used in the game. C1 keeps count of correct answers in each round, and is again initialized at line 960. C2 keeps count of the absolute number of problems generated, and is used for going from one round to the next. C3 keeps a cumulative count of the total number matched correctly on the first try.

Lines 490-510 read in the literal items, which are stored as a data array of word pairs in lines 2000-4135. Care has been taken to keep all three sets of words equal in size, although this is not a strict necessity. In any case, the line numbering system used (2000-3000-4000) allows for expansion without the necessity for renumbering.

Line 520 is the beginning of the loop of ten problems per round. Since the absolute total of problems encountered is incremented in C2 (line 620), lines 530 and 540 determine from which group of items the test item will be selected. Thus, if C2 is less than 10, control passes to line 590, which sets a variable, 'z', equal to a random number between 1 and 72. If the student passes successfully through the first round, the condition set at 530 will be met, and control will pass to the number generator for the next round (570), and so on.

Lines 550, 570 and 590 generate random integers in the proper ranges for the three parts of the game. The number generated corresponds to the subscript of the literal variable 'w$', against which the student input ('x$' in lines 640 and 670) will be matched. The parameters of the generator are determined by the size of the array and the manner in which it is divided. In this case, the integers will be generated to fall between 1 and 72, 73 and 144 and 145 and 216. The random number actually generated by the function 'rnd' is between 0 and 1, thus necessitating some arithmetic to bring it to the proper value.

Lines 600 and 610 set a flag on the subscript variable 'z', sending control back to line 530 if that number is encountered a second time.

Line 615 displays to the student one of the 72 words in each main section. The 'correct answer' is a string corresponding to the other half of the word pair in the data array. Since the integer generated at random may be either odd or even, the correct answer matching 'x$' will be either 'w$(z+1)' (odd) or 'w$(z-1)' (even). By using the integer function ('int') to determine oddness or evenness of 'z' (line 630), control is branched to either 650 or 675 for a match test.

If a match is made on the first try a congratulatory message is displayed (line 680), counters 1 and 3 are incremented and the loop continues (line 742 is the bottom).

If a match is not made on the first attempt, the program branches to line 730, which prints the equivalent in the target language of "the answer is." The variable 'z' is again evaluated for evenness or oddness, and an appropriate branch to either 735 or 739 gives the student the correct answer, followed by a request to type it. Unless the student input ('t$') at this point matches, control branches back to 730 and the loop continues until a match is made.

The remainder of the program is taken up with evaluation by section and score, and is for the most part straightforward.

This program can be used for vocabulary games in any language. The data-read feature will easily handle an array of a few hundred items, but for larger item arrays it would probably be advisable to use a data file.

```
list spiell

spiell      10:16   09/25/74  wednesday

70 print 'welcome to spiell, the German word game.'
80 print'do you want instructions?(yes/no)'
85 input a$
90 if a$= 'no' then 360
100 print'in this game you will have to match a word'
110 print'with its antonym or opposite meaning.'
120 print'for example, if you are given "hot" you'
130 print'are expected to type "cold." in'
140 print'this game,the words are in german.'
150 print'the entire game has three parts,'
160 print'going from easy words to harder words.'
165 print 'some items are more than one word'
170 print'on each part you will have to match seven out'
180 print'of ten in order to go on to the next part.'
190 print'if you fail to do this in the first or second parts'
200 print'you will automatically be put out of the game.'
210 print'the words are chosen at random from large'
220 print'pools of lexical items.in some cases'
230 print'a word or phrase may have more than one antonym,but'
240 print'only one of them has been arbitrarily chosen'
250 print'for the game.umlauts do not exist'
260 print'in the system, so you are asked to type'
270 print'the following for them:ae, oe, ue,aeu.'
271 print 'do not capitalize nouns.'
272 print'questions and comments should be directed'
274 print'to prof. barney milstein, stockton state'
276 print'college, pomona, nj 08240.'
280 print'the scale for the scores is as follows:'
290 print'                    15-20-------student'
300 print'                    21-22-------oberdummkopf'
310 print'                    23-24-------dummkopf'
320 print'                    25-26-------unterdummkopf'
330 print'                    27-28-------klug'
340 print'                    29   -------intelligent'
350 print'                    30----------LEHRER!!!'
360 print'give me a number to get me started,please'
365 input a
380 q = rnd(a)
400 dim w$(216),f(216)
410 c1=0
420 c2=0
430 c3=0
440 rem c1 ctr for corr ans;c2 for total in each section,
450 for 1=1 to 216                           c3 for tot crct.
460 f(1)=0
470 next 1
480 rem f(1) is for flagging used words
490 for i= 1 to 216
500 read w$(i)
510 next i
520 for j= 1 to 10
530 if c2<10 then 590
540 if c2<20 then 570
550 z= int(144+(70*rnd+1))
560 go to 600
570 z = int(72+(70*rnd+1))
580 go to 600
590 z = int(70*rnd+1)
600 if f(z)=1 then530
610 f(z) = 1
615 print w$(z)
620 c2 =c2+1
625 rem test for odd or even z
630 if int(z/2)=z/2 then 670
640 input x$
645 f(z+1)=1
650 if x$<>w$(z+1) then 730
660 go to 680
670 input x$
674 f(z-1)=1
675 if x$<>w$(z-1) then 730
680 print 'gut!!'
690 c1=c1+1
710 c3=c3+1
720 go to 742
730 print 'die antwort ist: ';
733 if int(z/2) = (z/2) then 739
735 print w$(z+1);'--type it, please.'
736 input t$
737 if t$ = w$(z+1) then 742
738 go to 730
739 print w$(z-1);'--type it please.'
740 input t$
741 if t$ <> w$(z-1) then 730
742 next j
750 if z<= 72 then 870
760 if z<=144 then 925
770 print 'your score for all three parts is' ; c3
780 print 'your rating is'
790 if c3=30 then 1000
800 if c3 =29 then 1020
805 if c3>=27 then 1040
810 if c3>=25 then 1060
820 if c3>=23 then 1080
830 if c3>=21 then 1100
860 if c3<20 then 1120
870 if c1<7 then 5000
880 print'you have done the first part'
890 print'with' ; c1 ;'correct answers'
900 print'you will now go on to the second part'
910 c1=0
920 go to 520
925 if c1<7 then 5000
930 print 'congratulations!!you have done the second part'
940 print 'with' ;c1; 'correctly answered'
950 print 'you will now go on to the third part.'
960 c1=0
970 go to 520
1000 print'LEHRER!!!'
1010 go to 1140
1020 print'intelligent'
1030 go to 1140
1040 print 'klug'
1050 go to 1170
1060 print'unterdummkopf'
1070 go to 1170
1080 print'dummkopf'
1090 go to 1170
1100 print'oberdummkopf'
1110 go to 1170
1120 print'student'
1130 go to 1170
1140 print'this is not so bad'
1150 print 'come back and play again some time. Auf Wiedersehen!!'
1160 go to5050
1170 print'you can do better.come back and'
1180 print'play again some time.auf Wiedersehen!!'
1190 go to 5050
2000 data 'bekannt','unbekannt','gut','schlecht','krank','gesund'
2010 data 'lang','kurz','gross','klein','schwach','stark','arm','reich'
2020 data 'frueh','spaet','klug','dumm','leicht','schwer','immer','nie'
2030 data 'ohne','mit','heiss','kalt','vermeidlich','unvermeidlich'
2040 data 'nichts','alles','dunkel','hell','links','rechts','dick','duenn'
2050 data 'froh','traurig','suess','bitter','alt','jung','kuehl','warm'
2060 data 'letzt','erst','mehr','weniger','sicher','unsicher'
2100 data 'das ende','der anfang','fuehren','folgen','leer','voll'
2110 data 'hier','dort','rueckwaerts','vorwaerts','zuerst','zuletzt'
2120 data 'der frieden','der krieg','morgen','gestern','lieben','hassen'
2130 data 'das licht','die finsternis','der morgen','der abend','der tag'
2135 data 'die nacht'
3000 data 'hoch','niedrig','schoen','haesslich','faul','fleissig'
3010 data 'fuer','gegen','ueber','unter','vor','hinter','messbar'
3020 data 'unmessbar','viel','wenig','weich','hart','feucht','trocken'
3030 data 'sichtbar','unsichtbar','falsch','wahr','guenstig','unguenstig'
3040 data 'tapfer','feig','locker','fest','nackt','gekleidet','oft','selten'
3050 data 'offen','geschlossen','oben','unten','scharf','dumpf'
3060 data 'geschmacksvoll','geschmacklos','genau','ungefaehr','laut','leise'
3070 data 'moeglich','unmoeglich','weit','nah'
3100 data 'beginnen','enden','tot','am leben','sprechen','schweigen'
3110 data 'loben','tadeln','glauben','zweifeln','frei','gebunden'
3120 data 'der freund','der feind','freundlich','feindlich','woher','wohin'
3130 data 'verschwinden','erscheinen','ueberall','nirgendwo','niemand'
3135 data 'jedermann'
4000 data 'einfach','kompliziert','besonders','allgemein','sorgfaeltig'
4010 data 'schlampig','aeusserlich','innerlich','eng','breit'
4020 data 'genuegend','unzulaenglich','ausserordentlich','gewoehnlich'
4030 data 'winzig','riesig','gerade','krumm','staendig','unterbrochen'
4040 data 'himmlisch','irdisch','vorher','nachher','ledig','verheiratet'
4050 data 'wachen','schlafen','einschlafen','aufwachen','bejahen'
4060 data 'verneinen','aufmachen','zumachen','geben','nehmen'
4070 data 'flach','huegelig','mitternacht','mittag','bestehen','vergehen'
4080 data 'schliessen','oeffnen','sterben','geboren werden','sinnvoll'
4090 data 'sinnlos','vergessen','sich erinnern'
4100 data 'stolz','demutig','anzuenden','ausloeschen','erlauben','verbieten'
4110 data 'fuellen','leeren','loesen','binden','sinken','schwimmen'
4120 data 'wecken','einschlaefern','hinstellen','aufheben'
4125 data 'ablehnen','annehmen'
4130 data 'der vorteil','der nachteil','sich anziehen','sich ausziehen'
4135 data 'einpacken','auspacken'
5000 print'you have gotten less than seven correct '
5005 if c2>10 then 5015
5010 print'on the first part'
5012 go to 5020
5015 print'on the second part'
5020 print'go home and try again another day'
5030 print'a u f'
5040 print'w i e d e r s e h e n !!!'
5050 end
```

Computer Cartoons

by Ronald E. Anderson and Marilyn Freimuth
University of Minnesota

People are generally not comfortable with computers. The rapid proliferation of computers has left many suffering mild anxiety from the social adjustments required in the wake of such "future shocks." Laughter has been known to cushion all types of shock and psychiatrists have even gone so far as to prescribe humor for people's distresses. For this reason, humor deserves a closer look in the context of the computer movement.

Humor communicates far more than happy emotions because it depends upon a collection of *ideas* which are not completely compatible. A funny cartoon or a good joke can provide a great deal of valid commentary about computers by calling attention to important facets of the real world, especially the beliefs and values that people hold. Consequently humor often expresses the complexities and subtleties of cultural thought and sentiment.

In the pursuit of such insight into the public's computer mood, we systematically began to collect computer humor.[1] Cartoons, especially single-frame drawings, are the primary form of humor on the subject of computers. Rarely can you find a computer mentioned in a joke book or a public restroom. Evidently this is due to the distance most people feel removing them from the computer and perhaps because the computer is a natural victim of caricature. Distortion of physical features constitutes the essence of caricature, e.g., Mr. Nixon is all cheek and nose whereas computers become forbidding monsters or smiling faces.

Creativity is an extremely important ingredient of both humor writing and humor enjoyment. Laughter depends upon the surprise revelation of a new, unexpected link between disjoint things or thoughts. Perceiving a fresh connection or an unexpected irony requires a creative leap from one established thought to another. Surprise yields mild tension; laughter is the human mode for relieving that tension. Without creative perception of the unexpected character of humorous incongruities, a human being experiences neither surprise nor pleasure.

It is our conviction that cartoons reveal people's attitudes toward computers more effectively than many other methods of assessing attitudes and feelings. In this spirit we compiled a library of cartoons from such diverse sources as magazines, newspapers, and books although most of the cartoons were taken from *The New Yorker, Saturday Review, The Saturday Evening Post, Punch, Datamation,* and *Educational Technology.* For these magazines we collected all the computer cartoons in five year intervals beginning in 1952 and ending in 1972, which provides a broad sampling of public mood across the early years of computer history. Our data base now contains nearly 300 cartoons, and these have been systematically subjected to content analysis to identify the dominant theme in each cartoon. This is not an easy task because some cartoons possess multiple themes, some of which are overlapping and very subtle.

Six major themes occur in the computer cartoons:

1. humanized computer
2. computerized human
3. computerization penetrating daily life
4. computer as beneficial tool
5. tool evolves into threatening master
6. the dependent computer
7. computer people and insider jokes

Each cartoon was placed into one of the above categories, and it is insightful to examine the cartoons in each category separately to gain a perspective on the ideas and orientations basic to each theme. As indicators of public attitudes the cartoons are considerably more elaborate and rich than opinion research studies. Nonetheless, the cartoon portrait is consistent with the opinion studies. In particular both types of investigations demonstrate the combination of both positive and negative feelings toward computers. Not only is there divergence of attitude orientation but some if not most people feel ambivalent toward computer-

1 Humor being one of the most creative of human activities should challenge the more creative computer artists, but as yet no computerist has created a program to generate jokes or gags. The senior author would be delighted to hear from someone who has proven our claim incorrect. Please address any responses to Ron Anderson, 2122 Riverside, Minneapolis, Minnesota 55404.

ization. To illustrate these findings each of the content categories is briefly summarized:

1. *Humanized computer*. A large portion of computer humor assigns human traits to computers. These cartoons are much more likely to appear in literary rather than technical magazines, which suggests that the human-computer identification problem is greater in the general public than in the computer world. The human-like computer cartoons attribute thinking, feeling, socializing, misbehavior, and even sex drives to computing machines.

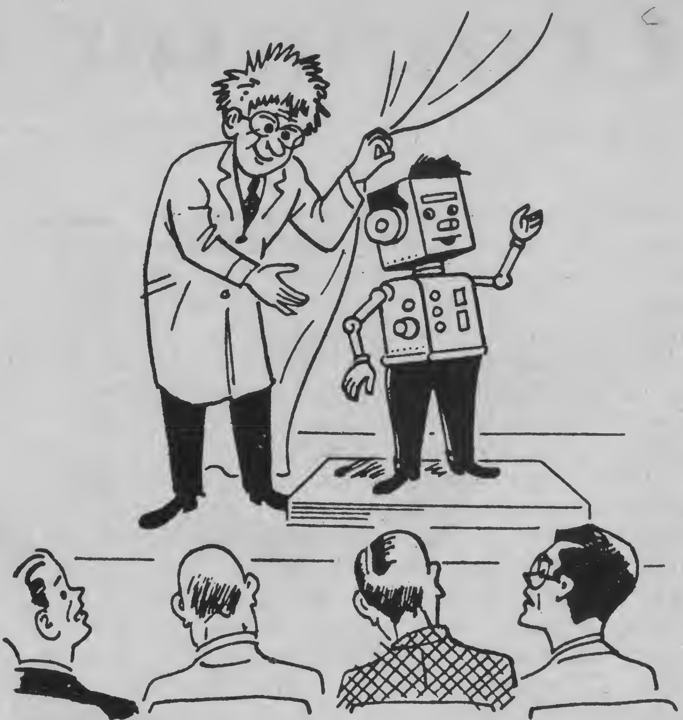
"Only once in every generation is there a computer that can write poetry like this."

© DATAMATION

The major preoccupation of the cartoons is the issue of whether or not computers think and to what extent computing resembles thinking. A more playful trend is the emphasis upon emotional problems, e.g., in 1957 (June 19), *Punch* shows two tired engineers belatedly report on the state of computer repair: "I'm afraid it needs a psychiatrist."

2. *Computerized human*. Several cartoons support the contention of social critics that our society has become robopathic and many people are losing personal spontaneity and compassion for fellow human beings. Sometimes the blame for this condition is attributed to computerization; sometimes not. Usually the people suffering from dehumanization are those working close to computers.

These cartoons can be viewed as a positive aid in identifying areas where people act like machines forgetting traits that make one uniquely human.

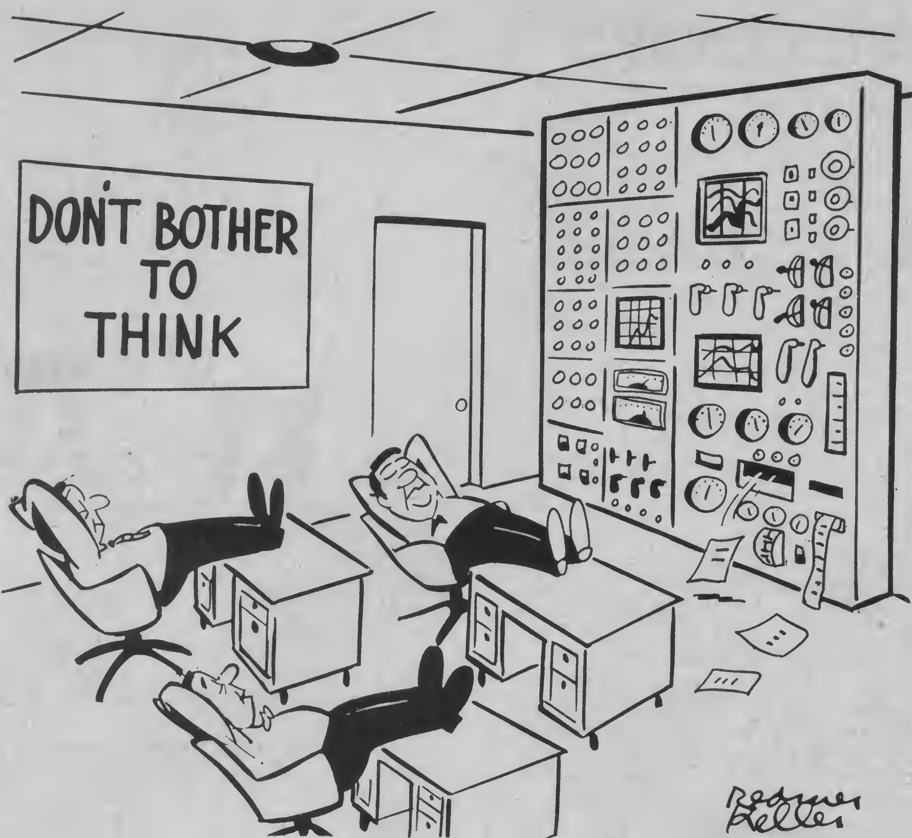


"... AND NOW, GENTLEMEN—MY GREATEST ACHIEVEMENT... THE PERFECT PUPIL!"

3. *Computerization Penetrating Daily Life*. The largest grouping of cartoons points toward this theme. Computers are intentionally or accidentally depicted as a normal, growing institution. Several social sectors are emphasized: education, business, and the family. Highlighting this type of humor are those focusing upon the intertwining of computers and social life in areas such as computer dating.

"Gee! my first computer date! I wonder what he'll be like?"

MODERN DATA



4. *Computer as beneficial tool*. This theme is, of course, the primary justification for the existence of the computer. Cartoons decorate this point by showing many idiosyncratic ways of putting the computer to work: as prophet, as fortune teller, as Santa's helper, as fishing expert, and so forth. One could surmise that man is coming to rely too heavily upon the computer.

5. *The Tool Evolves into Threatening Master*. Extreme reliance on any technology has its disadvantages. An appropriate metaphor for the theme in this area is the servant turned master. Heavy dependence on computers as pervasive tools may lead to obedience to the computer as an authority. This dilemma is depicted in a 1964 (June 3) *Punch* cartoon where an onlooking scientist observes another scientist bowing down in front of the computer saying, "I think Smith's on to something pretty big!"

6. *The dependent computer*. In a crucial way computers depend on human intervention for data and programs. Perhaps the most interesting idea which the cartoons elaborate on is that no matter how complex a computer may be, its operation is still subject to the whims and sporatic moods of people. The act of "pulling the plug" so that the computer goes down is reinterpreted as an act of self-actualization.

7. *Computer people and insider jokes*. The technical magazines tend to emphasize this area. Some humor resides with the computer itself and such problems as high cost, miniaturization, and hardware change. The more interesting cartoons focus upon computerpeople and popular images of this unique breed of specialist. Often the computerman is a superman, but more often he (she) is struggling to solve huge problems while on the brink of personal disaster.

Conclusion

Skimming through these cartoons and associated themes uncovers humor which quite blatantly raises social issues, e.g., impersonalization and unemployment. Other cartoons are more subtle in their message but nonetheless function as consciousness raising devices. They serve to sensitize the reader to the role of the computer in society.

Computer cartoons also offer us a creative, human way of coping with a technology that is sometimes frightening, sometimes boring, and sometimes incomprehensible. Cartoons provide us with a delightful route to think and feel about the most fantastic of man's machines.

CREATIVE COMPUTING
Feature Review

34 Books on BASIC

Stephen Barrat Gray
Gray Engineering Consultants
260 Noroton Ave.
Darien, Conn. 06820

The first book about BASIC was published on the first day of 1966. Four years later only eight more on BASIC had been published, and there was little difficulty in choosing between them, especially since several weren't very well known.

As the use of BASIC grew, so did the number of books on the subject, until today there are nearly three dozen and, for several years, up until 1974, they were being published at the rate of one every two months.

Choosing a book on BASIC is not so easy now, whether for one's own reading or for class use. The task is only slightly simplified, and only for certain prospective users, by these 34 books falling into several categories: two are very simple, for children in the lower grades; six are limited in coverage, meant to give a solid foundation in elementary BASIC; and three use BASIC as part of a book (or set of books) on the larger subject of data-processing. But that still leaves 23 to pick from. Hopefully, this "group review" will make the choice a little simpler.

The Reason Why

This article had its origins several years ago while, in addition to being the corporate EDP communicator, I was managing a small time-sharing installation. Interested in learning all I could about BASIC, I began to collect the various books on the subject. After getting a dozen together, the feeling grew that a group review of all such books might be of interest to those looking for one or two that would best suit their needs.

This group review is not a scholarly dissertation, but hopes to be of help to those looking for a BASIC text for study or for use in the classroom. If it fails in this, then perhaps it can be considered as a reasonably accurate chronological bibliography of all the books in English on BASIC.

THE AUTHOR

Stephen B. Gray first became acquainted with data-processing as a field-service engineer with IBM, after which he wrote maintenance manuals for airline-reservation computers at Teleregister (now Bunker-Ramo). Five years as the computers editor of *Electronics* magazine at McGraw-Hill were followed by several at General Electric, supervising the writing of manuals for the 115 and 130 computers. He next became editor of John Diebold's *ADP Newsletter,* and then editor of American-Standard's corporate EDP newsletter and manager of their time-sharing installation. He is now an EDP consultant and writer, and is the editor of a technical consumer magazine. In 1966 he founded the Amateur Computer Society, and publishes its newsletter for an international membership of people who are building a digital computer as a hobby.

The Authors

These 34 books were written by at least 52 authors, nearly all of whom are college professors, or teachers in private schools. As for the others (at the time of the books' publication), Albrecht (14, 28 32) is with Dymax (part of the People's Computer Company), Barnett (23) is with TRW Systems, Farina (3 13) with General Electric, Keenan (12) with the National Science Foundation, Sack (27) is at Amdahl Corp., Smith (10 30) is with Control Data, and Spencer (6) is president of Abacus Computer Corp. No affiliation is given for Pavlovich (18) or for Stern & Stern (34). Two of the authors are students: Kaufman (33) and Tahan (18). The authors of two books are unknown: General Electric (15) and NCTM (4).

A few authors, including Farina (3 13) and Sack & Meadows (27) dive headlong into BASIC and shout "Come on in, the water's fine." The majority swim with varying degrees of enthusiasm in waters of various depths and temperatures. Some stand on dry land, point to the pool, and say "It's over there." Gruenberger (25) and Hare (8) say, in effect, that although BASIC may be fine for schoolchildren, there's nothing for a real man like the strong surf of FORTRAN.

The biggest problem for many authors is an apparent inability to put themselves in the reader's shoes, and write for the average beginner. All too often a program is presented without enough previous discussion of the statements and the programming techniques involved to allow the reader to readily understand the program. This "too much too soon" problem is severe enough in several books to make some average readers simply give up in bewilderment. Many of these books seem to have been written with the top student in mind, by authors who either don't realize that most readers are starting at zero, or who seem to assume that the reader is as smart as the writer. Three books in point are by Smith (10 30) and Gruenberger (25), who give the textual impression of being eccentric geniuses; under firm editorial control, their considerable talents could have produced outstanding books, rather than fascinating curiosities, best opened after mastering one or two less convoluted texts.

Some will argue that many of these books need to be used in a classroom, with a teacher to explain the hard parts and to fill in the gaps. Indeed, some of the texts seem written with the expectation that someone will be on hand to do just that.

The Writing

Nearly all the authors are in the academic world. No doubt several are brilliant in front of a class. But little of this brilliance appears in the rather pedestrian prose of most of these texts.

Although it can be argued that these works are not novels and thus there is no point in trying to achieve any particular literary style, nevertheless there is quite a difference in wading through the still waters of some authors' works, and in dipping into the sparkling brook of Farina's writing (3, 13), the careful detailed prose of Kemeny & Kurtz (2), the enthusiasm and flair of Dwyer & Kaufman (33), or the clear, flowing style of Sack & Meadows (27).

Feature Review of Books on Basic

	Title	*Author*	*Publisher*
1.	BASIC, Sixth Edition	Waite and Mather	University Press of N. E.
2.	BASIC Programming	Kemeny and Kurtz	Wiley
3.	Programming in BASIC	Farina	Prentice Hall
4.	Introduction to an Algorithmic Language (BASIC)	(no author)	NCTM
5.	Introduction to Computing Through BASIC Language	Nolan	Holt, Rinehart & Winston
6.	A Guide to BASIC Programming	Spencer	Addison-Wesley
7.	Problem-Solving With The Computer	Sage	Entelek
8.	Introduction to Programming: A BASIC Approach	Hare	Harcourt-Brace
9.	BASIC For Beginners	Gateley and Bitter	McGraw-Hill
10.	Discovering BASIC	Smith	Hayden
11.	Basic BASIC	Coan	Hayden
12.	Computer Science: BASIC Language Programming	Forsythe, et al.	Wiley
13.	Elementary BASIC With Applications	Farina	Prentice-Hall
14.	Teach Yourself BASIC	Albrecht	Tecnica
15.	Time Sharing's BASIC Language		General Electric
16.	BASIC Programming	Murrill and Smith	Intext
17.	BASIC: An Introduction to Computer Programming . . .	Sharpe and Jacob	Free Press
18.	Computer Programming in BASIC	Pavlovich and Tahan	Holden-Day
19.	An Introduction to the BASIC Language	Skelton	Holt, Rinehart & Winston
20.	Basic BASIC: Self-Instructional Manual	Peluso, et al.	Addison-Wesley
21.	BASIC Programming for Business	Sass	Allyn & Bacon
22.	Fundamental Programming Concepts	Gross and Brainerd	Harper & Rowe
23.	Programming Time-Shared Computers in BASIC	Barnett	Wiley
24.	Introducing BASIC	Blakeslee	Educomp
25.	Computing with the BASIC Language	Gruenberger	Canfield Press
26.	Business Programming with BASIC	Diehr	Wiley
27.	Entering BASIC	Sack and Meadows	SRA
28.	My Computer Likes Me	Albrecht	Dymax
29.	Elements of BASIC	Lewis and Blakeley	NCC
30.	A Visual Approach to BASIC	Smith	CDC
31.	BASIC, A Computer Programming Language . . .	Pegels	Holden-Day
32.	BASIC	Albrecht, Finkel & Brown	Wiley
33.	A Guided Tour of Computer Programming in BASIC	Dwyer and Kaufman	Houghton Mifflin
34.	Principles of Data Processing	Stern and Stern	Wiley
35.	Learning BASIC Fast	DeRossi	Reston
36.	Programming in BASIC with Applications	Singer	McGraw-Hill
Review Copy Not Received –			
	Interactive Computing in BASIC	Sanderson	Butterworth
Not Reviewed –			
	BASIC In Ten Minutes	Hoitsma	Dartmouth
	School BASIC	Weissman	Dartmouth

The books are given an overall A-B-C-D rating. Ten are excellent, and rate an A. Ten more are very good, for a B. Another ten are average, C. Two are poor, getting a D. Two get split ratings: one gets an A for the first half of the book, a C for the second half; the other book rates a B+ for classroom use, but only a c- for the solitary reader.

Some authors chop their texts into very brief chapters, perhaps feeling that the reader's attention span may be very short for such material; Skelton (19), for example, has 12 chapters in 158 pages, with a 2½-page chapter on READ and DATA alone. Other cram quite a number of statements into each chapter, usually in related groups. Sass (21) has only nine chapters in 310 pages, and Barnett (23) has eight chapters in 366 pages.

Only eight of the books introduce string constants and variables at the same time as numeric constants and variables, rather than later in the text, or not at all.

Several of the authors are more terminal-oriented than others, and discuss the Teletype within the first half-dozen pages: Albrecht et al (32), Dymax (28), Pavlovich & Tahan (18), Pegels (31), Spencer (6) and Sass (21).

In ten of the books, INPUT is introduced before DATA, perhaps to emphasize the interactive nature of BASIC.

Several authors, or perhaps it was their publishers, have padded their books with a great deal of white space, blank pages, and meaningless appendixes. Several other authors, notably the anonymous ones of the 53-page NCTM booklet (4), have managed to cram more into each page than have the writers of many of the much longer books.

No Standardization

Comparing any two or three of these books with each other shows that there is no standardization in even the simplest features of BASIC. Most authors are divided between, for example, IF-THEN and IF/THEN; a couple use IF . . . THEN. Some authors write Basic, others BASIC. Waite & Mather (1) use the phrases "loop variable" and "running variable"; others call it the counter, index, control variable, index of a loop, or control identifier. Over half the books have no name for it at all, other than simply "I." Only Gross & Brainerd (22) distinguish between brackets and parentheses; Dwyer & Kaufman (33) also use both, but don't say why.

Three books mention one statement each that no others include: APPEND, PAUSE, and TYP.

The Programs

Most of the authors begin by presenting programs on a slow-and-easy basis, starting with no more than 2 to 5 lines, and building up from there. Kemeny & Kurtz (2) start with a 5-liner on page 1; NCTM (4) opens with a 2-liner. Gruenberger (25) is one of the few mavericks in this respect; he seems to believe in the sink-or-swim theory, and starts off with an 11-line program on page 2. Even though every line has a paragraph of explanation, the program is too difficult for many readers.

Farina, in his earlier book (3), is one of the few authors to keep all his programs short; the longest one contains 14 lines. The longest in NCTM (4) is 15 lines; in Peluso et al (2) it is 13, except for two longer ones; Stern & Stern (34) have only two over 15 lines; Gateley & Bitter (9), only two over 13 lines long. Even the 63-page Dymax (28) has programs 28 lines long.

A number of authors work their way up into some very long and overly complex programs, most of them too complicated for many beginners; these include Smith (10 30), Gross & Brainerd (22), Sharpe & Jacob (17), Nolan (5) and Hare (8).

Only a few authors go into the different levels of programming languages. Sack & Meadows (27) and Murrill & Smith (16) do a little, Hare (8) and Nolan (5) do more (2½ pages each), and Gross & Brainerd (22) do quite well by the subject, with 5½ pages.

Two authors discuss the history of time-sharing: Sass (21) and Spencer (6) take two pages each.

Sage (7) is the only author to use the expression "falling through" and to explain the principle thoroughly.

Most of the authors spread the teaching of the elements of BASIC over most of the length of their books. But there are some others who prefer to devote the major portion of their book to applications. For instance, Sage (7) has only 65 of 244 pages on the elements of BASIC; in Gross & Brainerd (22), BASIC is covered in the first 68 of the 304 pages; and Kemeny & Kurtz (2) spend only 43 of their 150 pages on the essentials.

There are a few contradictions between one book and another. For instance, Forsythe et al (12) have a section in the first chapter on testing a program, with a number of suggestions, including "When this experimental approach fails to reveal the trouble . . . . Another technique called *tracking* then becomes very helpful . . . . It is done by inserting PRINT statements at selected points in a program being tested." But Gruenberger (25) has a different viewpoint: "Warning: as a debugging tool, tracing is to be regarded as a last-ditch resort, and should never be used casually. Using a tool as crude as tracing for debugging is the mark of a poor programmer." That may well be true of an experienced programmer, but a beginner needs all the help he can get.

Only four texts show concern for the esthetics of programming. Blakeslee (24) puts it one way, "SERMON: Always remember the poor sucker who has to use the output of a program you write; keep it neat, keep it simple. That poor sucker could be you!" Gruenberger (25) notes that allowing a program to end in OUT OF DATA ON LINE XXX is "not the most graceful way to terminate a program." Sharpe & Jacob (17) say that a printout without headings is "hardly every elegant output." Waite & Mather (1), practical as always, not only note that a program ending in OUT OF DATA "does not yield an attractive printout," but add that it "prevents taking any action *after* the program discovers that it has run out of data."

Also in the realm of esthetics, although more on the side of readibility, are the suggestions in five of the books to use blank lines to "divide visually the major sections of a program," as Waite & Mather (1) put it. Seven books indent the statements inside a loop, between a FOR-NEXT pair.

Flowcharts, Indexes, and Tutorials

Not many authors are big on flowcharts; Kemeny & Kurtz (2), for instance, have only three in the whole book. Sage (7) and Smith (10) have flowcharts for every program example, and Peluso et al (2) have for most of their programs. Sass (21) has the most complex flowcharts, with Smith (10) not far behind. Others who make frequent use of flowcharts are Coan (11), Nolan (5), Lewis & Blakeley (29) and, above all, Forsythe et al (12), to whom flowcharting is everything.

Few authors seem to understand the art of indexmanship, as most have only a few pages; Barnett (23), for instance, has five pages of index. The one exception is Hare (8), whose index is a full 18 pages long. Four books have no index: Dymax (28), General Electric (15), NCTM (4) and Smith (10).

A few indexes must have been computer-generated, because they have some references to subjects that are mentioned only very briefly in the text, and which nobody would probably ever want to look up. The Gross & Brainerd (22) index lists "coconut" and "animal, carnivorous," while Kemeny & Kurtz (2) list "Oz, Land of."

Hare also has the longest glossary: 16 pages. Nolan (5) has 11½ pages, Sass (21) has 8, and Spencer (6) has five pages of glossary.

Seven of the books have authors who think enough of their programs to have separate indexes of them, by title and page number.

Most texts assume that the reader knows enough about the various areas of mathematics to need no tutoring, but several others devote sizable numbers of pages to teaching math, including nine pages on matrices and ten on trigonometry in Pavlovich and Tahan (18).

more

Exercises and Problems

All the books have exercises (some call them problems) that require writing programs, except for the General Electric programmed-instruction text (15). The "self-instruction" text by Peluso et al (20) has question-and-answer exercises, plus an appendix containing practice problems that require programs to be written. A few other texts require a minimum of program-writing. Skelton (19) has exercises that require modifying given programs and writing short subroutines. Waite & Mather (1), being a user's manual, has no problems or exercises of any kind.

There is a wide variety of ways of presenting problems and answers. Some books, such as Sharpe & Jacob (17) have problems at the end of each chapter, with all answers given. Sack & Meadows (27) give answers to selected problems at the end of each chapter. Kemeny & Kurtz (2) give no answers to the end-of-chapter problems.

Many authors provide exercises after each new idea (or group of new ideas); among these, Coan (11) gives answers to the even-numbered exercises, Sass (21) gives answers to selected ones, and Gross & Brainerd (22) give none.

A few books have only questions and answers, such as the General Electric text (15). There are very many Q&A in Stern & Stern (34) and all too many, 396 of them, in Diehr (26). Some readers like a good many questions, to help reinforce their learning; others will prefer more text or more programming problems and fewer questions.

Although most of the authors provide problems or exercises that are quite satisfactory, most of these are straightforward mathematics problems. This is fine for most readers, but for the younger ones, and perhaps also for those of us who get bored easily, several authors have out of their way to provide problems of unusual interest. Albrecht et al (32) have problems, for example, on generating "computer art," Barnett (23) on computing the (x y, z) position of a satellite, Dwyer & Kaufman (33) on an airline reservation system, and Kemeny & Kurtz (2) on writing four-part harmony for a given melody.

Handsome Is . . .

Not every good-looking BASIC book is also good reading. On one hand, the three handsomest single-volume works – Kemeny & Kurtz (2), Gateley & Bitter (9), and Barnett (23) – rate highly as useful taxes in addition to having excellent typography and layout. Also to be listed among the good-looking books are those by Sharpe & Jacobs (17), Pavlovich & Tahan (18), Gross & Brainerd (22), and Dwyer & Kaufman (33).

But the two handsome multiple-volume sets – Forsythe et al (12) and Stern & Stern (34) – have BASIC supplements that rate poorly, although Stern & Stern's main text and workbook give an excellent coverage of data processing.

Any bound book is difficult to use comfortably at the Teletype or other terminal. The first version of the earlier Smith book (10) was loose-leaf, so that individual pages (of heavy stock) could be removed for use at a terminal. The problem there, of course, was that those removable pages could be lost.

Most of the books use several type styles, but some overdo it, with up to seven in some cases, so that many pages are very distracting, as in the books by Ivan Flores. Hare (8), Murrill & Smith (16) and Albrecht et al (32) use up to seven typefaces on some pages, Spencer (6) has up to six, and Dymax (28) and Gruenberger (25) have as many as five. Dwyer & Kaufman (33) also use as many as seven, but in a way that isn't anywhere nearly as obtrusive as the others, which is one more tribute to the fine design of this book.

Half the authors show programs in Teletype originals (and many should have put a new ribbon on the machine); the rest use typed or machine-set examples. Two of the authors – Smith (10) and Coan (11) – give examples to problems requiring programs, in the back of the book, in Teletype originals reduced so greatly as to be eye-straining. Lewis & Blakeley (29), for some reason, show their Teletyped programs much smaller than necessary, even though there is plenty of room for them to be shown much bigger.

The size of each book is given because the number of pages alone is deceiving. Many of the smaller books contain over 300 pages, but contain less text than some larger books with many less pages. The most popular size is 6 by 9 inches, with 15 books at or near those dimensions. Second most popular is 8½ by 11 inches, with a dozen that size.

Eight of the authors are sinners in the eyes of Teletype Corp., because they write "teletype." No doubt all have been cautioned by Teletype's eagle-eyed legal department.

Personal Preferences

If I were limited to choosing only one book from these 34, it would be Kemeny & Kurtz (2), which is still the standard of excellence by which all the others must be judged. Waite & Mather (1) give all the nitty-gritty details. Then come, in order of preference, Barnett (23), Murrill & Smith (16), Dwyer & Kaufman (33), and Gross & Brainerd (22).

If a friend were to ask for a recommendation, it would be, again, Kemeny & Kurtz (2), if he wanted only to learn BASIC. For someone wanting to know about computers and BASIC, then it's Hare (8). If he wanted to learn about BASIC and business programming, I'd recommend Stern & Stern (34) for their main text and workbook, plus one of the five BASIC books listed above.

For any young person, or as a matter of fact for almost anybody, I'd recommend Dwyer & Kaufman (33), who do their best to make learning fun.

For a future group review of books on applications of BASIC, I would appreciate information concerning such publications. This would include not only books such as Peckham's *Computers, BASIC and Physics*, but also applications books not oriented toward any particular language, but which could be used with BASIC, such as Gruenberger & Gaffrey's *Problems for Computer Solution*.

Also appreciated is information about books on BASIC in languages other than English.

1. *BASIC, Sixth Edition*, edited by Stephen V. F. Waite and Diane G. Mather. In print Mar. 19, 1971, University Press of New England, Hanover, N. H., 183 pages, 8½ x 11, $4.00 (paperback).

The first BASIC user's manual, and still the best, although it describes an advanced version of BASIC. Rating: A

The first few editions of this user's manual were authored by John G. Kemeny and Thomas E. Kurtz, the originators of BASIC; the more recent editions carry the names of the editors.

According to Prof. Kurtz, the first draft of the BASIC instruction manual appeared in June 1964. The third edition was published on Jan. 1, 1966. ("Whatever happened to the second edition, I'll never know. I *think* we started counting editions with number three, and chose that number just to be on the safe side.") The date of the fourth edition is Jan. 1, 1968; Version I of the supplement, Feb. 28, 1969; Version II, April 3, 1969. The preliminary fifth edition, 1969, the fifth edition, 1970. The sixth edition, 1971; second printing, 1972.

According to the British book, *Specification for Standard BASIC*, by Bull, Freeman and Garland (National Computing Centre, London, 1973), "the first implementation was on a GE 265 system in 1964. The first issue of the programming manual from Dartmouth College (hitherto this and subsequent updates were the only definitive documents on BASIC) was in January 1965 . . . By January 1966, the third edition of the manual was published."

Of all the 34 books, this is the only user's manual on BASIC, with all the details, enough to satisfy the most inquisitive time-sharer. However, many of the advanced statements will be unknown and useless to any reader who does not have access to one of the five systems now using BASIC VI: Dartmouth, Annapolis (the Naval Academy), Computer Sharing Services in Denver, Grumman Data Systems on Long Island, and Polycom Systems in Toronto. Nevertheless, it is still valuable and fascinating, not only for its complete description of "standard" BASIC, but for showing us what can be done with some very interesting (and tantalizing) extensions to the standard language. It shows how extensive and powerful BASIC can be, especially when one reads the sections on FILES and segmentation.

The book is printed from typed originals, but is so crammed with valuable information that the reader easily overlooks the difficulty, if any, of reading typed pages. The programs are reproduced from Teletype originals.

There are ten chapters: introduction, BASIC primer (12 statements, loops), more about BASIC, files, segmentation, arrays, the TEACH system, debugging and compiling, error messages and other information (ASCII character set, accuracy and timing considerations), and a summary of BASIC.

The book starts right off with a 10-line program, a simple one on bank balance, with two and a half pages of explanation. There are four BANK programs, each one expanding and improving on the previous one, building up to a 33-line program by page 14. There are also five blank lines in BANK-4, "to divide visually the major sections of the program" and "greatly enhance the appearance and readibility of the program." Only four other books do this: Kemeny & Kurtz (2), Barnett (23), Diehr (26) and Dymax (28). Some program lines are indented, mostly those inside a loop.

The authors say "The technique of ending a program by having it run out of data is very simple and efficient. However, it does not yield an attractive printout and prevents taking any action *after* the program discovers that it has run out of data." Several other authors make note of the esthetic point, but none remark on the practical point.

The book abounds with unique nuggets that no other authors mention. "If the FROM, TO, and STEP elements of the FOR statement form an impossible combination . . . the body of the loop will not be performed and the computer will proceed immediately to the statement following the next." Another: "A step size of zero . . . in any loop where a positive step size is needed will cause an infinite loop."

The book is the only one that tells how to overprint, by using the carriage-return character, CH$(13).

There is a long, full and excellent description of FILES, 25 pages of it. There are 15 pages on error messages, some of which are very intriguing, such as ILLEGAL TREE CLIMBING, OUT OF ROOM, and UNDERFLOW.

As examples of the goodies available with this version of BASIC, there are eight special characters for defining PRINT USING fields or areas where variables are to be printed, and twelve commands for debugging, such as BREAK and TRACE.

The negative points are few and far between. Where every other system uses RESTORE, this one uses RESET. No examples are given to show the results of simple MAT operations. And no example is given of TEACH, an instructor's test program, although the chapter on the TEACH system is 4½ pages long. Very little information is given about RND, only three-quarters of a page.

This book, then, is excellent as a reference, or for reading if you want to know all there is to know about BASIC; BASIC VI, that is. Or to read after working with BASIC awhile, as this book will tell a beginner much more than he may want to know. The book covers (or so it seems) every last possible detail, and has a highly authoritative ring to it, which is only natural. Perhaps because of its extensive treatment of the language, the books, several authors and editors seem to have more of a sense of the "big picture" than all the others.

* * *

BASIC: An Inttroduction to Computer Programming Using the BASIC Language, by William F. Sharpe. Pub. Aug. 1967, by The Free Press, div. of Macmillan, New York, N. Y., 137 pages, 6¾ x 10, paperback.

(Out of print, superseded by a revised edition, by Sharpe & Jacob (17).)

Sharpe published the first BASIC text. As he recounted recently, Sharpe had gotten an early user's manual and the specs from Dartmouth, and wrote a Fortran IV compiler for batch-mode BASIC, called UWBIC (University of Washington BASIC Interpretive Compiler). For a text to use in his classes, he wrote this book, and sent the manuscript to eight publishers. All eight said it was nice and well done, but only four showed an interest in publishing it. The other four said there was no market for a book on BASIC.

* * *

2. *BASIC Programming,* by John G. Kemeny and Thomas E. Kurtz. Second edition pub. June 18, 1971 (first edition pub. Oct. 20, 1967), by John Wiley & Sons, New York, N. Y., 150 pages, 8½ x 11, $7.75 (paperback).

Not the first text, but the best, on almost all counts. Rating: A+

A winner when it first came out, often imitated but only partially equalled (and seldom, even that), this book has been improved and enlarged in its second edition.

The modest authors make no reference to having been the originators of BASIC, although the publisher does so on the back cover. Even though Kemeny and Kurtz may be said to have an inside track, the excellence of this book is due rather to the authors' "simple, gradual introduction to computer programming and to the use of time-sharing systems," as the back cover puts it, plus the most careful attention to every detail, covering all the bases and leaving as few questions unanswered as possible. Although there are many fine features, the outstanding one is the immense care taken to ensure that the reader will have a minimum of difficulty in learning BASIC.

Several BASIC books have summaries of the statements on the inside cover; this is the earliest of three with examples of each included.

The preface spells out the background requirements for the various portions of the book: chapters 8 to 12 and 18 "may be mastered with a background of three years of high school mathematics." Chapters 15 to 17 "consider three mathematical areas [statistics, vectors and matrices, calculus] that are normally taught at the college level."

The first chapter (numbered zero because it is new to this edition) is a simple introduction, with a few paragraphs on what is a computer, what is a program, what is BASIC, and how a computer is used.

The first page of Chapter One, on Elementary BASIC, presents a five-line program that divides one constant by another. This is explained in four paragraphs. The second program is 17 lines long, converts meters and centimeters to feet and inches. Over a page and a half of explanation follow (and these are large pages), covering every detail more thoroughly than any other author on any program in any of these books; the runner-up is the Waite & Mather book (1), which was originally written by Kemeny & Kurtz. This second program uses the INT statement, which most authors don't introduce until later; it is explained neatly and completely in four sentences.

There is a short summary at the end of each chapter, followed by a dozen or so exercises that are quite sensible, and some even include hints. But there are no answers.

After the elementary chapter is one on Time Sharing, covering What is Time Sharing, commands, interaction in

BASIC, editing and correcting. Two sentences describe the purpose of an executive program, while other authors usually make a big deal out of it (although very few others even mention it).

The subject of loops is considered important enough to be worth the entire third chapter, five pages. Here the book begins to get a little difficult for those who are not mathematically oriented, with a program that computes binomial coefficients. The authors "use the convention of indenting instructions between a FOR-NEXT pair," which only six other books do. "This clearly shows the scope of a loop. The convention is particularly useful when nested loops occur, such as the double loop in ROOTS." Double loops are then illustrated with a neat 9-line program.

Page 19 contains the only example in this book of an author tooting his own horn, ". . . and then determines the cell of the tally list N in a very ingenious way in line 150."

In the fourth chapter, on Lists and Tables, DIM is introduced very casually, without fanfare or a long-winded lead-in, as is often the case elsewhere.

In chapter five on Functions and Subroutines there are no individual examples, although there are several short programs that include six of the ten standard functions listed.

The unique and excellent method of demonstrating rounding-off goes through each part of the argument of the INT statement to show how it affects the number involved. This is not an easy concept for many readers to understand, yet the authors have managed to find what must be the simplest way of explaining it.

Page 29 contains one of the few chinks in the armor: a program not explained. The reader is expected to know Euclid's Algorithm for finding the greatest common divisor of two integers.

Page 36 contains the most complex program so far, an Eternal Calendar, 54 lines, but this is not impossible for one without a head for math to figure out.

The chapter on Debugging goes into tracing, and is the only book to describe both "full trace" and "selective trace."

Part Two, on applications, begins on page 47, with chapters on problems from elementary mathematics, number theory, simulation, games, business, files, text processing, statistics, vectors and matrices, calculus, and "special topics."

The chapter on number theory is where the book really begins to get difficult for all but math majors. The problem on the greatest common divisor isn't all that difficult, but the next one is, on modular arithmetic, using the Chinese remainder theorem. The rest of the chapter isn't any easier, with a counting problem (making change) being rather difficult, even with the explanation, which could be longer, but perhaps would have to be much too long for non-math-majors.

The next chapter, on simulation, eases off somewhat, and contains an explanation of RND that is simple and easy to grasp, and is better done than elsewhere. The chapter contains a baseball program that simulates the batting of one side in a nine-inning game; this is rather complex for all but the top-IQ types. The Knight's Tour problem is also rather difficult for a beginner's book.

With the chapter on Business Problems, the book goes back to something less than difficult. Many other authors would put those complex chapters (on number theory, simulation, and games) at the end of the book, so as not to discourage a reader who is only halfway through.

The chapter on Files is the only one among these books that distinguishes between Teletype files and numeric-and-string files, with a compact little table to summarize the differences.

The chapter on Statistics goes into contingency tables, using the chi-square test; both are introduced nicely, understandable even to those not majoring in mathematics, although there is no explanation of "number of degrees of freedom." The cahpter also contains a section giving A Ranking Procedure, with an explanation that is rather murky.

The text of the chapter on Vectors and Matrices is a little too loose to be readily understandable; the text accompanying the problem on electrical networks is not at all understandable to other than electrical engineers; the section on Markov Chains is not very well explained and both runs give data that is difficult to identify.

The chapter on Calculus is for math majors only, or for those with a knowledge of advanced math.

All the applications chapters contain, after the exercises at the end of each chapter, one or more projects that present more complex problems than do the exercises. These are all well thought out, and should provide the reader who has access to the terminal with a very thorough workout of his knowledge of the language, as well as, in many cases, his knowledge of (and aptitude for) mathematics.

The last section of the book is on Harmony in Music, giving a long program that writes four-part harmony for a given melody. The three projects for this chapter involve preparing a program to generate simple melodies randomly, generating poetry randomly, and devising programs to produce artistic patterns "on whatever plotting devices are available," with a number of helpful hints.

The program index at the end of the book is unique: for each of the 107 programs in the book, it gives the name, application, and page. The main index must have been computer-generated, as there are several trivial entries, including Ramanujan and "Oz, Land of."

As for drawbacks, the main one is that there are no answers to the exercises. The book contains only three flowcharts. The reader with only a modest mathematical background may have difficulty with the chapters on number theory, simulation, and games. REM is seldom used, although the authors say in a footnote that "REM statements are not always used in the programs in this book because all the programs are amply discussed in the text."

There are very few individual examples of statements and commands in this text; however, many sample programs make up for this by showing the statements and commands in use.

All in all, this is the best book on the subject at this level, even better in its second edition than the first.

The second edition inserts a five-line program as the first one in the book, before the 17-line one, to further simplify the beginning. These programs, and all the others in the book, were rerun for the second edition. A few blank lines have been added to separate groups of statements in programs not previously so treated.

The chapter on timesharing has been moved up from fifth to second place, and increased from two and a half pages to over four. Also added is a section on Commands in Time Sharing, and information on eight editing commands. The chapter on functions and subroutines has a new section on standard functions and multiple-line definitions.

The chapter on loops opens the same, but a new program has been inserted as the first one, on permutations and combinations. Several new exercises have been added at the end of the chapter, and some of the others have been reworded; this holds also for some other chapters.

The sixth chapter, on Alphabetic Information, is new to the second edition, and is all about strings. The first chapter on applications drops the section on Large-Number Arithmetic, which is a good idea, as it required a long and complex program. Replacing it is the Counting Problem, on making change, which is much easier to understand and to appreciate. The chapter on business problems adds a section on Criticial Path Analysis.

The two new chapters in the applications part of the book are on Files and on Text Processing (line editing, character handling, constructing an index, codes).

The last chapter, Special Topics, eliminates the section on Teaching Machines, along with its teaching program. In its place is the complicated Marriage Rules in Primitive Societies, not as interesting but perhaps more indicative of what is being run on computers these days. The section on A Model From Ecology is the same, with one interesting change. In the first edition, the program output is a graph, plotted on a Teletype. In the second edition, the graph is made by an X-Y plotter. Three projects have been added at the end of this chapter, using the computer to generate melodies, poetry, and artistic patterns.

* * *

3. *Programming in BASIC, The Time-Sharing Language*, by Mario V. Farina. Pub. Feb. 1968, by Prentice-Hall, Englewood Cliffs, N. J., 164 pages, 8½ x 11, $7.50 (paperback).

Slow but sure approach, in one of the best of the elementary texts. Rating: A

On one hand, the preface states that the book is intended for engineers and computer-programming students, as well as "programmers who need an easy-to-use language for checking out programs written in a more difficult programming language."

On the other hand, the book is designed like a primer, with short paragraphs of only one to three sentences, spaced a line apart, which spreads out the material and makes it so easy to read that this would be an excellent book for secondary schools.

The opening is on a high-school level: "Do you have problems? Do those problems involve repetitive calculations using a desk calculator or a slide rule? Why not have a computer help solve your problems? It's easy!"

The first program has five lines, multiplies one constant by another and adds a third to that. Widely spaced, the five lines take 1 5/8 inches of vertical space, the most "spaced-out" of any of these books. The programs are all in the same type as the text.

There are simple exercises at the end of each lesson, with "answers to selected exercises" at the back of the book.

The approach is slow but sure, with everything covered, leaving little if anything to the imagination. This is one of the best of the elementary BASIC texts, by a born teacher whose later book (13) has an even better style.

Each chapter takes quite a few pages to present a limited amount of material. Lesson 1 has a five-line program and a long explanation of it, then goes into legal names. Lesson 2, on What Is BASIC, is about writing equations, using LET. Lesson 3 goes into Teletype time-sharing, covering commands and shows a drawing of the keys and buttons of a model 33 Teletype. Lesson 4 is on flowcharting and is the longest chapter in the book, 14 pages that proceed very slowly and carefully, covering all the bases.

The style is conversational and light: "Of course, any names you choose would be OK, but names chosen should remind you what they stand for." There is a great deal of white space, such as half an inch of it above and below examples of program lines.

Lessons 5 through 11 are on: telling the computer about numbers, exponential notation, telling the computer what to do with numbers, built-in functions, making decisions, having the computer print out answers, and arrays and subscripts. Lessons 12, 13 and 14 are all on loops, a total of 24 pages.

Lesson 14, on Loops Within Loops, is 14 pages of the most explicit of all these books on this particular subject, and also goes into double-subscripted arrays, using the "loop-within-loop idea to set to 50 all elements of a double-subscripted array." This chapter is about as complicated as the book gets. Lesson 15, on Matrix Computations, has no examples of what MAT statements actually do to a matrix. This is the only chapter that could perhaps be improved, by having such examples, although their omission is in keeping with the secondary-school level of the text.

Lessons 16 through 18 are on home-made functions, subroutines, and INPUT. The lesson on Home-Made Functions, which is about DEF, shows how to use it to define hyperbolic sine and cosine, which is surprising in what up to this point was such a simple book.

Lesson 19, on Library, is about stored programs, and lists eleven that were among the 50 available on the General Electric system at the time of writing, including TRUINT*** and BLKJAK***.

Lessons 20 through 22 are on paper-tape usage, making corrections, and system commands and features.

Lesson 23, on Edit Commands, is on delete, extract, merge, weave and resequence. Lesson 24, on Extended BASIC Features, goes into strings, RANDOMIZE, multiple assignments with LET, TAB, MAT INPUT, passwords, etc.

Lesson 25 presents "A Program From Beginning to END," showing a 13-line program as typed in by the user, then the corrections made, the improvements and further errors and corrections, a total of six pages that are unique and valuable.

The Appendix gives a summary of BASIC statements.

One of the few drawbacks is that, although there are 43 programs in the book, there are no runs at all, except for the "Program From Beginning to End" in the last lesson, and three tiny, one-line printouts in the next-to-last chapter, on Extended BASIC Features. The reader may develop a somewhat constipated feeling, having to digest all these programs that have no output.

* * *

4. *Introduction to An Algorithmic Language (BASIC).* Pub. May 1, 1968 (third edition, 1972), by National Council of Teachers of Mathematics, 1906 Association Drive, Reston, Va. 22091, 53 pages, 6 x 9, $1.40 (paperback).

Excellent for what it sets out to do. Rating: B+

According to the introduction, "This booklet aims to help the mathematics teacher introduce computers through an easy, problem-oriented language." It achieves this goal, giving the essentials of BASIC in as little space as possible. It is a fine illustration of what can be done well in a minimum of space, packing much more information per page than many longer books, without skimping on detail.

The booklet starts right off on page 2 with a two-line program on squaring. The subsequent eight programs elaborate on that, all in Section I, on Introducing BASIC. In Section II, Sample Problems and Exercises, the first problem is on finding divisors of a positive integer, with three more programs that develop this further. The next program is on maximizing an area, with two elaborations; the last is on mean and standard deviation (this by page 32!), with one program 13 lines long. Each program introduces new concepts and statements, with an explanation of each program line where required.

There are three exercises at the end of the first section, and a total of eight within the second section. The answers are in the back of the booklet, in full; 8½ pages of answers for the eleven exercises, with 14 programs.

A short epilogue has a few words about other languages, and flowcharting.

Only nine statements are used (PRINT, END, READ, DATA, GO TO, INPUT, LET, IF-THEN, FOR-NEXT) and one system command, RUN. Nothing on REM, functions, matrices, arrays, lists, tables, etc.

If one must find fault, it is simply that this booklet is so well written that one wishes it were longer. Hence the highly subjective rating of B+. Many of those who find the booklet long enough would rate it A.

* * *

"Like the Roman god Janus who faces both ways, BASIC faces the needs of those just entering the world of automated data processing as well as those departing for its more stratified plateaus."

From the preface to Entering BASIC, *by John Sack and Judith Meadows (27)*

To be continued next issue.

Computer Course for Educators

Abacus Computer Corporation from time to time offers to secondary school teachers and educational administrators a comprehensive two day course of computer instruction. The courses are presented at various geographic locations. The course objectives are to provide tools and methods for: 1) learning about computers and programming, 2) reviewing other computer science programs in Secondary Schools, 3) developing an educational program for use in the attendees' own school and 4) determining current status of computer technology as related to education. It is planned that attendees will leave the course with a good exposure to computer technology and will understand how to create their own programs in the BASIC programming language.

A descriptive brochure concerning the course "MINICOMPUTERS IN THE CLASSROOM" may be obtained by writing or calling ABACUS COMPUTER CORPORATION, 194 E. Granada Ave., Suite 222, Ormond Beach, Florida, (904) 672-5672.

Send for their catalog of beginning computer science textbooks too. Mention *Creative Computing*.

JAN-FEB CAT POSTER

The description of the system which produced the CAT centerfold in the Jan-Feb issue was written by Sam Harbison, Princeton Univ. It comes to us courtesy of Lee Ratzen, College of Medicine & Dentistry of NJ and Marilyn Spencer, Ridgewood High School.

The Computer Picture system is a technique for reproducing photographs on the computer line printer. It is useful not only for creating artistically pleasing pictures in an unusual medium, but for various scientific applications.

The actual translation process involves many steps. First, the photographic negative is converted to a computer-readable form by a scanning digital densitometer. This device measures the optical density of the film at a number of points and writes this information on a magnetic tape. The number of points scanned on the negative is variable: the user may select a grid spacing of 25, 50, or 100 microns. The data on the magnetic tape is used as input to a series of computer programs which crop the picture on the tape and translate the optical density values (which range from 0 to 255) to a single printer character, or a combination of characters which are overstruck, to achieve the desired shading. Since there are many fewer printer shadings that density levels (typically 17 different shadings are used on the printer), there is much leeway in the mapping from the tape data to printer shadings, which enables one to produce many interesting effects.

The expansion of the one point on the negative (50 microns square, say) to the size of one printer character (.1 inch square) represents a linear expansion factor of about 50. 35mm negatives scanned at 50 microns will be about 4x6 feet when printed.

A number of pictures of general interest have been produced via the Computer Picture system and can be obtained at minimal cost at the Princeton University Computer Center or from one of the remote terminals. A list of available pictures and the job needed to obtain copies of the pictures are available in the Clinic at the Computer Center.

The Computer Picture system was designed and implemented by Sam Harbison at the Princeton University Computer Center Clinic with the assistance of Prof. Robert langridge and the Department of Biochemical Sciences in the use of their scanning digital densitometer. Thanks also go to the staff of the Princeton University Computer Center Clinic, who provided programming advice and the funds used to develop the system.

NEW INSTRUCTIONAL COMPUTING BOOKLET

The ACM Special Interest Group on Computer Uses in Education has sponsored a special 80-page publication, entitled *Topics in Instructional Computing* devoted to teacher education in instructional uses of computing. Edited by Stuart Milner of the School of Education at Catholic University of America, *Topics* contains twelve refereed papers which discuss methodology, languages, resources (materials, organizations, etc.), attitudes, effective learning environments, and familiarizing teachers at all levels with the broad spectrum of instructional computer uses.

Copies of *Topics in Instructional Computing* are available at $4.00 each from A. Kent Morton, Kiewit Computation Center, Dartmouth College, Hanover, NH 03755.

Can Computers fall in love?

Do computers have a sex? Does a computer built under Scorpio get along with a programmer who was born under Capricorn? Could a computer conspiracy ever arise? Could you live a daydream through a computer?

Use subscription form on Page 4.

If you've ever thought about these questions before, or if you're first thinking about them now, then it's time you thought about "Creative Computing"—the magazine that speaks *your* language.

"Creative Computing" is a bi-monthly publication that's about everything that computers are about. From computer poetry to computer art. From the effects of computers on pollution to their effects on privacy. From computers as crime fighters to computers as teaching aids.

"Creative Computing" gives you the chance to be a matador in a bull fight, govern the ancient city of Sumaria, and even fight a space war. Those are only a sample of the kinds of computer games you'll find. Or how about some non-computer games and puzzles?

And that's not all. "Creative Computing" has book reviews, cartoons, fiction, and even a fold-out poster. Plus news and commentary on the twenty computer education projects that have endorsed this publication.

So get involved in the curious world of computers now. Subscribe to "Creative Computing". It's *the* magazine for *the* curious mind.

CREATIVE COMPUTING
P. O. Box 789-M
Morristown, New Jersey 07960

BULK RATE
U. S. POSTAGE
PAID
PERMIT No. 153
Morristown, NJ

[redacted]

ATTN: Computer Science Dept.

Third Class