Light Pens And Graphics Tablets: How To Use Them

COMPUTE!

$2.95
May
1984
Issue 48
Vol. 6, No. 5
£2.25 UK $3.25 Canada
02493
ISSN 0194-347X

The Leading Magazine Of Home, Educational, And Recreational Computing

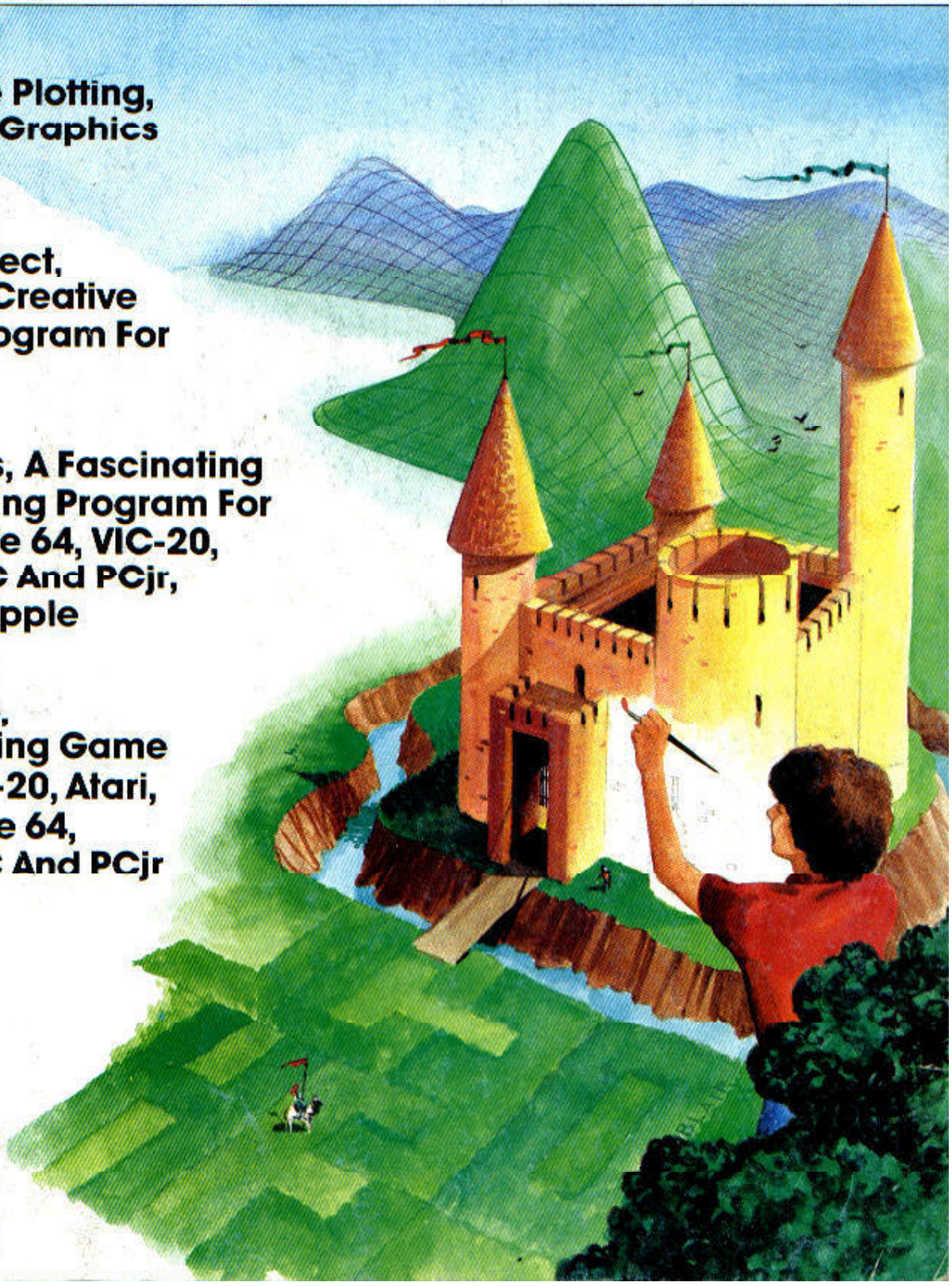
3-D Surface Plotting, A Valuable Graphics Technique

Picture Perfect, A Friendly, Creative Drawing Program For Youngsters

Pentominos, A Fascinating Puzzle-Solving Program For Commodore 64, VIC-20, TI-99/4A, PC And PCjr, Atari, And Apple

Plus Snertle, A Challenging Game For The VIC-20, Atari, Commodore 64, And IBM PC And PCjr

And More



0 71486 02193 05

May 1984 Vol. 6, No. 5

FEATURES

EDUCATION AND RECREATION

REVIEWS

COLUMNS AND DEPARTMENTS

THE JOURNAL

NOTE: See page 179 before typing in programs.

GUIDE TO ARTICLES AND PROGRAMS

The Digital Palette: Fundamentals Of Computer Graphics	*
Light Pens And Graphics Tablets: New Ways To Communicate With Your Computer	*
The Inside Story: How Graphics Tablets And Light Pens Work	*
Realtime Dreaming With Mike Newman	*
3-D Plotting	64/AT/AP/PC/PCjr
Picture Perfect For Atari And Commodore 64	AT/64
64 Hi-Res Graphics Editor	64
Snertle	64/AT/TI/C/AP/PC/PCjr
Pitstop	AT/64/AD
Panic Button For VIC And TRS-80 Color Computer	V/C
The Editor's Notes	*
Readers' Feedback	*
Questions Beginners Ask	*
Computers And Society: Computers In The Workplace	*
On The Road With Fred D'Ignazio: The Morning After, Part 1	*
Learning With Computers: Ready-To-Run Magazines	*
The Beginner's Page: A Random Leap	*
INSIGHT: Atari	AT
Programming The TI: File Processing, Part 3	TI
Machine Language: A Program Critique, Part 2	*
64 Explorer	64
Pentominos: A Puzzle-Solving Program	PN/64/AT/PC/PCjr/TI/C/AP
BASIC Style—Program Evolution	*
VIC/64 Memdata	V/64
A BASIC Cross-Reference	P/64
Atari Softkey	AT
Atari Line Check Utility	AT
Commodore Word Wizard	V/64/P

AP Apple **AT** Atari, **P** PET/CBM, **V** VIC-20, **C** Radio Shack Color Computer, **64** Commodore 64, **TS** Timex/Sinclair, **TI** Texas Instruments, **PCjr** IBM PCjr, **PC** IBM PC, **AD** Coleco Adam, *All or several of the above.

COMPUTE! Publications, Inc. abc
One of the ABC Publishing Companies

One of the ABC Publishing Companies:
ABC Publishing, President, Robert G. Burton
1330 Avenue of the Americas, New York, New York 10019

COMPUTE! The Journal for Progressive Computing (USPS: 537250) is published monthly by COMPUTE! Publications, Inc., P.O. Box 5406, Greensboro, NC 27403 USA. Phone: (919) 275-9809. Editorial Offices are located at 324 West Wendover Avenue, Greensboro, NC 27408. Domestic Subscriptions: 12 issues, $24. Send subscription orders or change of address (P.O. form 3579) to **COMPUTE!** Magazine, P.O. Box 914, Farmingdale, NY 11737. Second class postage paid at Greensboro, NC 27403 and additional mailing offices.

ISSN 0194-357X.

READERS' FEEDBACK

The Editors and Readers of COMPUTE!

Can Disks Be Mailed?

Should disks be mailed, and if so, what is the proper way to mail them?

Brian Mangan

Disks can be mailed, as long as they are enclosed in a snugly fitting, rigid package. Many office supply stores sell padded jackets (called mailers) especially made for 5¼-inch disks. Also, for what it's worth, many users write a message on the outside of the mailer, to warn mail handlers that the package contains a magnetic recording which can be damaged by electromagnetic fields.

Commodore Sequential Append

I recently made a discovery that I think will help programmers using Commodore disk drives to create and use sequential files. In addition to writing a sequential file (OPEN 2,8,2,"SEQFILE,S,W") and reading a sequential file (OPEN 2,8,2, "SEQFILE,S,R"), it is possible to append a sequential file. This is a great help; rather than having to rewrite the entire file when additions are made (OPEN 2,8,2,"@0:SEQFILE,S,W"), all you have to do is use an A in place of the W when you open the sequential file for writing: OPEN 2,8,2, "SEQFILE,S,A". The DOS finds the end of the file and simply adds on the new data. You use the regular PRINT#2 statement to accomplish this.

Steve Gibson

Disabling The Atari Break Key

I want to inform your readers about a technique I discovered that disables the Atari's BREAK key, but does not need to be reexecuted after each GRAPHICS command. It is so simple that I wonder why no one has ever mentioned it, or if it conflicts with something that I have not yet found out:

POKE 566,143:POKE 567,231 to disable

and

POKE 566,84:POKE 567,231 to enable

The preceding statements change the BREAK key interrupt vector to point to address 59279 ($E73F) which contains a machine code PLA and RTI instruction used by the OS. This method will work only with the OS B ROMs, which contain the interrupt vector for the BREAK key.

Neil Weisenfeld

A TI Quit Fix

Have you ever hit FUNCTION + instead of SHIFT + while you are typing in a program? It's extremely frustrating to see all your work go down the drain. Here's a way to disable the QUIT key on the TI.

To do this you will need either the Mini Memory or Editor/Assembler cartridge or Extended BASIC and the 32K Memory Expansion. This is because the console BASIC does not contain the CALL LOAD subprogram (better known as POKE). Whenever you turn your computer on, type the following line in the command mode: CALL LOAD(–31806,16). This will disable the QUIT key. If you are using Extended BASIC, use CALL INIT::CALL LOAD(–31806,16). If you wish to return to the Master Title Screen, you can still do so by typing BYE.

Credit for this information goes to the documentation that comes with the TI Forth package.

By the way, does anybody know of a comprehensive memory map for the TI?

Davin A. Trulsen, Jr.

What's An EPROM?

I would like to know what EPROMs are and what they are used for.

Bob Cullen

EPROM stands for Erasable Programmable Read Only Memory. EPROMs are memory chips which can "remember" programs even when the computer's power is switched off. Important machine language programs like the BASIC language or the computer's operating system are often permanently stored in ROM, but standard ROM can be programmed only once (when the chip is made). EPROMs, on the other hand, can be programmed by any computer user with a relatively simple peripheral device, the EPROM programmer. EPROMs can also be erased by exposing them to ultraviolet light. You could use an EPROM to store any machine language program you use frequently—even to make your own game cartridges.

64 Sprite Collisions

I have a Commodore 64, and am having trouble with collision detection with sprite graphics. I use the following line to check for collisions:

```
IF (PEEK(53278)ANDX) = X THEN action
```

This is easy to convert to machine language. In all of my programs, this statement is unreliable. Sometimes it detects a collision between two sprites when they aren't colliding, other times it doesn't detect a collision when they are touching, and other times it works just fine.

I've read in past articles that this problem may be caused by "sparkle" on the 64, and that the solution to the problem is to relocate screen memory. I tried that, and it didn't help.

I've also found that by putting a PRINT PEEK(53279) in my programs, the collision registers work every time. But I don't know how to PRINT a PEEK in machine language.

Eric Rotenberg

First, sparkle can cause spurious collisions with sprites, but you have to relocate the character set, not the screen, to disable the sparkle. Second, be aware of the nature of the collision register. It is set when two sprites collide, and stays set, even after the sprites have moved away from each other.

Also, the register is cleared when you try to read it, so you can't keep doing an LDA or a PEEK to check for different collisions. The first PEEK resets the register. If the sprites are still touching, they will then set the collision register again. When you are checking for a collision, save the results of the first PEEK for later use.

BASIC B For The Atari 400 And 800?

1. Is Atari going to make a Revision B of BASIC, as found in the new XL series on cartridge or other form for the 400 and 800 computers?

2. I've been having trouble with my BASIC cartridge. *Pac-Man* works just fine, but when I plug in BASIC, either the screen goes blank, or I get two clicks and the screen goes blank, or it goes right into memo pad mode. This happens after I put in any other cartridge. Can anyone help me?

Kevin Bailey

As far as we know, Atari has no plans for offering an upgraded BASIC.

Even though ROMs are sturdy, solid-state devices, they can be damaged by static electricity or by being dropped. It's a good idea to ground yourself (by touching something made of metal) before you operate any computer equipment. But your BASIC's not necessarily bad. You may just need to clean the contacts.

Normally, the contacts are not exposed, but you can stick a pencil or paper clip into the slot to lower the protective hood. Then, using a swab and rubbing alcohol, thoroughly clean the contacts, then let the cartridge dry. Incidentally, this is also a recommended procedure for your Operating System board and other RAM boards. You may also want to try some TV tuner cleaner in place of the rubbing alcohol.

We don't know of any problems with one cartridge leaving the machine in a state that prevents it from running another cartridge, especially since the power is cut off between cartridge changes. If any other readers are having similar problems, or have a cure, please write in.

Slow TI BASIC

In his review of *Robot Runner* for the TI-99/4A in COMPUTE!, January 1984, Tony Roberts stated that games written in BASIC on the TI are notoriously slow because the microprocessor can't interpret BASIC fast enough. I want to clear up any implication that the TMS9900 CPU in the 99/4A is at fault.

TI BASIC is indeed slow, due to the unusual architecture of the machine and the design of the BASIC interpreter. First of all, the RAM in which BASIC programs are stored is not CPU RAM. The 16K of RAM in the 99/4A is maintained by the TMS9918A video display processor (VDP). There are only 256 bytes of CPU RAM in the 99/4A console.

Every time the microprocessor accesses or RUNs a BASIC program, it must request the program from the VDP one byte at a time, one statement at a time. This causes a great increase in execution time, because the microprocessor must wait for the VDP. While the TMS9900 microprocessor is a word-oriented (16 bits) chip, the VDP works in bytes.

The second reason why TI BASIC is so slow is that the interpreter itself is not written in machine language. It is written in another high-level language known as Graphics Programming Language, or GPL. The GPL interpreter is also built into the 99/4A console. Thus, whenever a BASIC program is RUN, a *double interpretation* takes place. This is similar to writing a BASIC interpreter in BASIC for an IBM PC. It is really amazing that the TMS9900 can run BASIC as fast as it does, considering.

Chris Clark

Use Of COMPUTE! Programs

Concerning the "Readers' Feedback" of September 1983, you stated that the programs in COMPUTE! are not in the public domain, and that only people who own a specific issue of COMPUTE! can have access to the programs in that issue. My question is, what if a computer club takes out a

subscription to COMPUTE!? Would that club be allowed to place those programs in those issues in its library for all members? And what if a school or public library takes out a subscription? Could everyone who is allowed access to the library be allowed access to those programs in those issues?

Gary Lee Crowell

Sorry, the answer in each case is no. You can only use the programs in an issue of COMPUTE! *if you own a copy of that issue.*

VIC Video Typewriter

I have written a short program that transforms your VIC into a typewriter (without any annoying syntax errors). I use it to practice my typing after school. To disable the program, use the f1 key.

Vicky Cwiertnie

```
10 PRINTCHR$(8):PRINTCHR$(14)
20 POKE36879,26:PRINT"{CLR}"
30 PRINT"** VIDEO TYPEWRITER **"
40 GETA$:IFA$=""THEN40
50 IFA$="{F1}"THENEND
60 IFA$=CHR$(13)THENPOKE36878,15:POKE3687
   6,220:FORX=1TO50:NEXT:POKE36876,0
70 PRINTA$;:GOTO40
```

Atari Tape Verify

Here is a one-line program which verifies that an Atari tape file is recorded properly. The utility works whether you CSAVE, LIST, or PRINT (data) to the tape. It performs essentially the same as Michael J. Barkan's "Atari Verify" (COMPUTE!, August 1983), but is much shorter. This utility can be LISTed to tape and ENTERed from tape, but since it is so short, it is easy to enter it from the keyboard in direct mode (without the line number). Just use this line:

```
0 CLOSE #1:OPEN #1,4,0,"C:":FOR A=1 TO
  400:GET #1,A:NEXT A
```

After recording a file on tape and while the program or data is still in memory, enter and run this utility. Rewind the tape to the beginning of the file and push PLAY. The utility will read the entire file, one character at a time, to insure that the file is recorded properly. Operation will end with an error code. If you get this code, the file was read successfully, showing that it is good:

```
136  END OF FILE
```

If you get one of the following error codes, save the file again, since it could not be read by the computer:

```
138  DEVICE TIMEOUT
140  SERIAL BUS ERROR
143  DATA FRAME CHECKSUM ERROR
```

The same variable is used for loop control and to hold each character as it is read from tape. This way, the loop never ends and will check any length of file. This variable can be changed to one of those in your program, if desired, to avoid adding to the Variable Name Table of your program.

Douglas J. Wilder

TI Randomness Test

Richard Mansfield's article "Zones Of Unpredictability, Part 2" ("The Beginner's Page," COMPUTE!, December 1983) included a program called "Randomness Test." Since it wouldn't work on my TI-99/4A, I wrote a similar program. It takes several thousand cycles to get close to even distribution for each number, but it's fun to let it run.

Gaston Porterie

```
100 CALL CLEAR
110 PRINT "TEST OF THE RANDOM NUMBE
    R","FUNCTION ON THE TI-99":::::
    :::
120 PRINT "PLEASE WAIT..."
130 T=T+1
140 RANDOMIZE
150 X=INT(10*RND)+1
160 A(X)=A(X)+1
170 FOR I=1 TO 10
180 P(I)=INT(A(I)/T*100)
190 NEXT I
200 IF T/100<>INT(T/100)THEN 130
210 CALL CLEAR
220 PRINT "AFTER";T;"CYCLES";"OF RA
    NDOMIZATION"
230 PRINT
240 PRINT "RANDOM","%","NUMBERS","O
    CCURRENCE"
250 S=0
260 FOR I=1 TO 10
270 PRINT I,P(I)
280 S=S+P(I)
290 NEXT I
300 PRINT "","---"
310 PRINT "TOTAL",S;"%"
320 GOTO 130
```

Easy DATA Statements

Here is a one-liner that I have found very useful while programming many statements that are almost identical. Used in the direct mode it can yield a set of DATA statements that fill the screen. The program can just as easily use POKE, or REM statements, or any combination of these.

```
FOR X=100 TO 300 STEP 10:PRINT X "DATA":
  NEXT X
```

Chuck Cole

Constant 1541 Errors

Ever since I bought my 1541 disk drive, I have been getting the errors 23 READ ERROR and 27 READ ERROR. This not only happens on my

disks, but also on prepackaged disks. I have read what these errors mean in Appendix B of my disk users guide, but these descriptions don't tell me much.

Could you please give me more information on these errors, and tell me what I can do about them?

Jay Elmore

The fact that this occurs both on your own disks and on commercial disk programs strongly indicates a hardware problem. Ask the dealer from whom you purchased the drive for the address of the nearest service center and have the drive checked out.

Sprite Data Problems

I am a Commodore 64 owner and I have a question about sprites. I understand how to create a sprite and move it around the screen. I also know how to move more than one sprite, if the data for them is the same. My problem occurs when I have more than one set of data. I can't seem to get both sprites on the screen at the same time. The *Programmer's Reference Guide* doesn't have an example with two sets of data. I would appreciate it if you would help me out.

Seth Hausman

Jim Butterfield replies:
I can think of two possible problems with your sprites:

1. *You may have forgotten to link each sprite to its drawing in memory. With normal memory mapping, sprite 0 needs to have its drawing number (usually 11, 13, 14, or 15) placed into memory address 2040, sprite 1 into 2041, and so on up to sprite 7 into address 2047. If you use drawing number 11, the drawing of the sprite should be in addresses 704–766 decimal; for number 13, addresses 832–894; for number 14, addresses 896–958; and for 15, 960–1022.*

2. *Many sprite register addresses control all eight sprites at the same time. To turn sprite 0 on, you would POKE 53269,1; to turn sprite 1 on, you would POKE 53269,2;* to turn them both on, *you would add 1 and 2 and POKE 53269,3. The following table shows the bit values for each sprite:*

Sprite	0 – 1
	1 – 2
	2 – 4
	3 – 8
	4 – 16
	5 – 32
	6 – 64
	7 – 128

Thus, to turn on sprites 0, 2, and 4, we add 1 + 4 + 16 and POKE 53269,21.

Be sure that you keep the difference between a sprite number and a drawing number clear in your mind. Several sprites can use one drawing (or "definition"); or a single sprite can be switched from one drawing to another as it moves its arms, legs, tentacles, or whatever.

Using Atari Cartridge Memory

I have an Atari 800, and am currently writing a text adventure game using the Assembler Editor cartridge. I hope to run the program without the cartridge when I'm finished. How can I use the 8K block of RAM used by cartridge (not to mention all those zero-page pointers that the cartridge uses)? Does it have to go to waste? I hope not, because I'll need all the memory I can get for this thing.

John Bushakra

No, the memory need not be wasted, but you cannot test the program with the Assembler Editor. Just define the memory you need, then assemble your program to disk. The object code will not go into memory, but will become an executable object file on the disk. The syntax is:

```
ASM,,#D:filename
```

You can then take all the cartridges out of your machine, boot DOS, then Load Binary File. If you make these the last two lines of your machine code

```
* = $2E0
.WOR START
```

where START is a label for the start address, your program will run automatically after it is RUN. Otherwise, you'll have to use Run At Address to start your program from DOS.

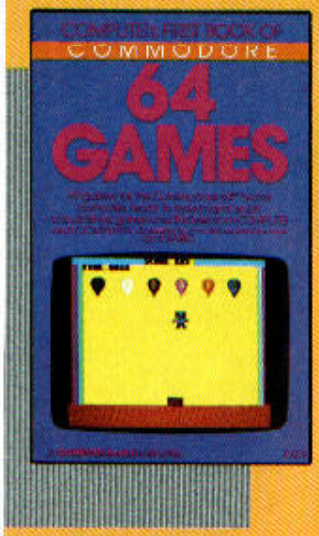
More Solutions For TI Cartridge Loading Problems

In the January 1984 "Readers' Feedback," I read a question about TI-99/4A cartridge loading problems. The problem was with lockup of the keyboard and broken screen display patterns after insertion of a program cartridge. The remedy given by COMPUTE! was to clean the contact strips of the program cartridge. I've found this to help, yet also discovered that this is not necessarily the complete solution. The cartridge connector extension that protrudes from the main circuit board may also be at fault. To remedy the problem means disassembling the computer, cleaning the contacts on both sides and both ends of the cartridge connector extension. This solved all of the problems I had encountered.

Richard Winslow

About four months after buying my TI, I had the same problem with loading the cartridges. I solved the problem by taking apart the computer and straightening the bracket which the cartridge plugs into. (It was bent.) Works perfectly now.

David L. Jones

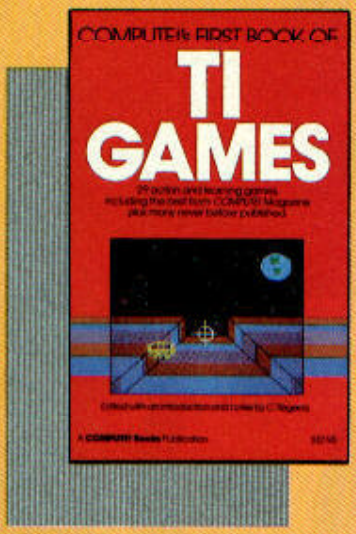


COMPUTE!'s First Book Of Commodore 64 Games

Packed full of games: "Snake Escape," "Oil Tycoon," "Laser Gunner," "Zuider Zee," and many more. Machine language games requiring fast hands and a good eye, as well as strategy games which will exercise your mind. Introductory chapters and annotated listings provide ideas and techniques for writing games. An excellent oduction for 64 owners who want to begin writing games.

pages, paperback.
al bound for easy access to programs.

2.95
10-942386-34-5

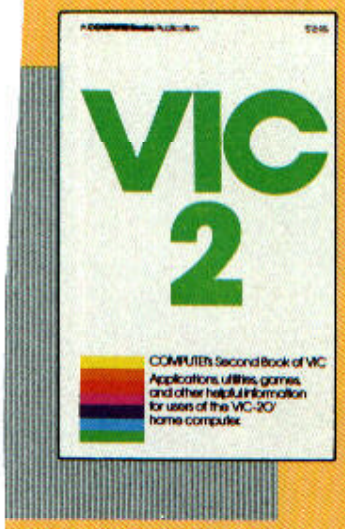


COMPUTE!'s First Book Of TI Games

Although this book is packed with ready-to-type in games (29 in all), it is more than just a book of games. It is designed to teach game programming techniques. Introductory chapters explain the special features of the TI-99/4 and 99/4A, giving advice on coding techniques. Most games include an explanation of how the program works. Contains mazes, chase games, old favorites, thinking games, creative challenges, and more.

211 pages, paperback.
Spiral bound for easy access to programs.

$12.95
ISBN 0-942386-17-5

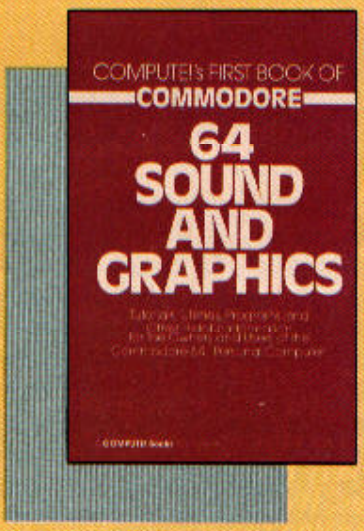


COMPUTE!'s Second Book Of VIC

This is just the book to follow the best-selling *First Book of VIC*: clear explanations of programming techniques, an extensive memory map, a mini word processor, a system for creating sound effects, a custom character maker, a machine language assembler, and "Gumball," an extraordinary all-machine-language game.

274 pages, paperback.
piral bound for easy access to programs.

12.95
BN 0-942386-16-7



COMPUTE!'s First Book Of 64 Sound And Graphics

Clear explanations of the 64's sound and graphics capabilities. Includes many tutorials and example programs: "MusicMaster," a complete music synthesizer; "High-Resolution Sketchpad," an all-machine-language program for making computer art; and "Ultrafont Character Editor," one of the best character editors available. The appendices feature useful reference charts and conversion tables.

275 pages, paperback.
Spiral bound for easy access to programs.

$12.95
ISBN 0-942386-21-3

COMPUTE!™ Publications, Inc. abc
One of the ABC Publishing Companies

Post Office Box 5406, Greensboro, North Carolina 27403

Snertle

Soori Sivakumaran

By making simple selections from a menu, a child can change this arithmetic drill to fit his or her own tutoring needs. Written for the unexpanded VIC, versions also are included for the Commodore 64, Atari, TI-99/4A, Color Computer, Apple, IBM PC, and PCjr.

"Snertle" is designed to help teach children the fundamentals of addition, subtraction, and multiplication. A turtle named Snertle is drawn on the screen to give encouragement and assistance to the player.

An Individual Challenge

Snertle allows children to tailor math problems to fit their individual abilities and weaknesses. Snertle first asks the child to select addition, subtraction, or multiplication problems. If addition or subtraction is selected, the child is then asked to choose the largest and smallest numbers to be used in creating the problems. The largest number that can be chosen is 99 and the smallest number is zero.

If multiplication is chosen, the child can decide to practice a certain "times table," or solve problems created randomly from 0 through the 14 times table.

For example, if the 12 times table is selected, then one number in each question created will always be 12. The other number will be randomly selected from the range 0–14.

If the child chooses to attempt random multiplication problems, he or she must define the range of numbers (within the limits of 0 and 14) from which the problems can be created, similar to the process for random addition or subtraction problems.

Creating The Screen

In Program 1, once the necessary information is entered, the turtle's image is POKEd onto the screen. The two numbers used in the problem are chosen in lines 305, 315, and 1070. The numbers are then displayed on the screen, each digit being four regular characters high and three wide. The large character set is created in a series of subroutines in lines 500–990.

The larger number is always displayed above the smaller number to avoid negative answers to subtraction problems. The appropriate sign for addition, subtraction, or multiplication is drawn on the screen by a subroutine beginning at line 6000. Next, a horizontal line is drawn under the numbers.

Line 394 contains a FOR–NEXT loop that clears the keyboard buffer. This prevents the child from accidentally entering data while the turtle and the problem are being put on the screen.

Another FOR–NEXT loop in lines 395–420 enters the user's response to the problem. Because a GET statement is used, the RETURN key does not have to be pressed when entering the response. An arrow will appear at the bottom of the screen to prompt for each digit of the response.

The Turtle Smiles

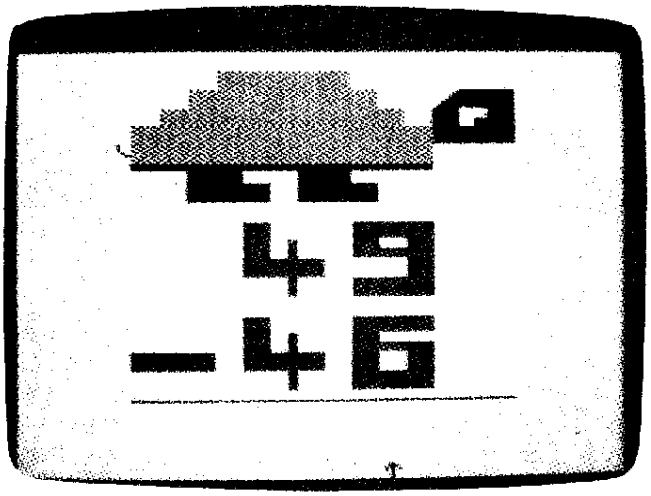
Once the response is entered, Snertle checks it against the correct answer. If the child's response is correct the turtle will smile, GOOD! will appear on its shell, and a high beep will sound. If the

response is incorrect, Snertle the turtle's head will disappear into his shell and the message TRY AGAIN will appear on his side.

The user will be given a second chance. If the new response is correct, Snertle will poke his head out from his shell. If the answer is again incorrect, the correct answer will be displayed on the screen.

The program will keep producing problems until the X key is pressed in response to a problem. The percentage of correctly answered questions is then calculated in line 410, and displayed. This percentage only includes problems answered correctly on the first attempt. Snertle then returns to the menu where the child may END the program or select more problems.

Program 1 uses all but 84 bytes of the unexpanded VIC's memory.



A subtraction problem—"Snertle" for VIC. Other versions similar.

Program 1: Snertle For VIC

Refer to the "Automatic Proofreader" article before typing this program in.

```
100 A$=CHR$(147):B$=CHR$(17):C$=CHR$(29):
    D$=CHR$(18):E$=CHR$(146):Y=160:LL=368
    76                                  :rem 62
110 PRINTA$SPC(5)B$B$"**SNERTLE**":POKELL
    +2,15                               :rem 181
120 PRINTB$B$B$B$C$C$ D$"SELECT ONE:"E$
                                        :rem 119
130 PRINTB$"1) ADDITION"                :rem 113
140 PRINTB$"2) SUBTRACTION"             :rem 117
150 PRINTB$"3) MULTIPLICATION"          :rem 87
155 PRINTB$"4) END PROGRAM"             :rem 30
160 PRINTB$"(ENTER 1,2,3 OR 4)";:INPUTQ:I
    FQ>4ORQ<ØTHEN160                    :rem 102
185 C=14:IFQ=1ORQ=2THENC=99             :rem 141
187 IFQ=3THEN1000                       :rem 224
188 IFQ=4THENEND                        :rem 248
190 PRINTA$B$B$"ENTER LARGEST VALUE"
                                        :rem 169
200 PRINT"(MIN.:Ø MAX.:";C;")";:INPUTR:IF
    R<ØORR>CTHEN200                     :rem 142
230 PRINTB$B$"ENTER SMALLEST VALUE"
                                        :rem 146
240 PRINT"(MIN.:Ø MAX.:";R;")";:INPUTS:IF
    S<ØORS>RTHEN240                     :rem 183
263 PRINTA$B$"PRESS "D$"X"E$" RETURN TO M
    ENU":FORI=1TO750:NEXTI               :rem 6
265 PRINTA$                             :rem 143
270 Z=Ø:ZZ=Ø:GOSUB2000                   :rem 55
275 GOSUB1100:GOSUB1170:GOSUB1230:GOSUB12
    60                                  :rem 102
301 TR=Ø:ZZ=ZZ+1                        :rem 226
305 L=INT(RND(1)*(R-S+1))+S             :rem 234
310 IFQ=3ANDT=1THEN320                   :rem 61
315 K=INT(RND(1)*(R-S+1))+S             :rem 234
320 F$=STR$(K):W=Ø                      :rem 243
325 IFK<LTHENW=110                       :rem 81
330 GOSUB3000                           :rem 217
335 W=110                               :rem 193
337 IFL>KTHENW=Ø                        :rem 244
340 F$=STR$(L)                          :rem 248
345 GOSUB3000                           :rem 223
346 ONQGOSUB6000,6000,6004              :rem 185
350 IFQ=1THENM=K+L                       :rem 97
355 IFQ=2ANDK>=LTHENM=K-L                :rem 78
360 IFQ=2ANDK<LTHENM=L-K                 :rem 11
365 IFQ=3THENM=K*L                      :rem 104
380 GOSUB740:MM=1:IFM>9THENMM=2         :rem 189
385 IFM>99THENMM=3                      :rem 101
390 GOSUB740                            :rem 183
393 V=Ø:GOSUB1100                       :rem 222
394 FORI=631TO640:POKEI,Ø:NEXTI         :rem 180
395 FORJ=Ø TO MM-1                      :rem 218
397 POKE8177-(4*J),30                    :rem 94
400 GETH$                               :rem 224
405 IFH$=""THEN400                      :rem 216
407 IFH$="X"ANDZZ=1THEN100               :rem 36
410 IFH$="X"THENPRINTA$"PERCENTAGE:";INT(
    Z/(ZZ-1)*100):GOTO120                :rem 10
412 FORO=8164TO8168:POKEO,32:NEXTO
                                        :rem 104
415 P=VAL(H$)                           :rem 199
420 V=V+(P*10↑J):X=8110-(4*J):GOSUB480:NE
    XTJ                                  :rem 86
450 IFM=VTHEN470                        :rem 210
451 POKELL,160:FORI=1TO500:NEXTI:POKELL,Ø
                                         :rem 83
452 FORI=8098TO8186:POKEI,32:NEXTI:rem 96
456 IFTR=1THEN460                        :rem 11
458 TR=1:GOSUB1500:GOSUB770:GOTO393
                                        :rem 159
460 M$=STR$(M)                           :rem 3
461 FORI=1TO22-MM:READA:NEXTI            :rem 96
462 FOROO=1TOMM                         :rem 204
464 P=VAL(MID$(M$,(OO+1),1))            :rem 243
465 READX:GOSUB480:NEXTOO:RESTORE:rem 222
470 GOSUB1230:IFTR=ØTHENGOSUB2500:GOSUB75
    5:Z=Z+1:GOSUB6500                   :rem 154
471 GOSUB2225:GOTO301                   :rem 238
480 IFP=ØTHENGOSUB720                    :rem 48
485 ONPGOSUB 500,525,555,585,610,633,660,
    680,700:RETURN                      :rem 254
500 FORI=ØTO66STEP22:POKEX+I+1,Y:NEXTI:RE
    TURN                                :rem 211
525 GOSUB990:GOSUB980:POKEX+44,Y:GOSUB970
    :RETURN                             :rem 102
555 GOSUB990:GOSUB980:POKEX+46,Y:GOSUB970
    :RETURN                             :rem 107
585 POKEX,Y:POKEX+22,160                :rem 193
595 FORI=44TO46:POKEI+X,Y:NEXTI           :rem 1
600 POKEX+23,118:POKEX+67,118:RETURN
                                        :rem 172
610 GOSUB990                            :rem 185
```

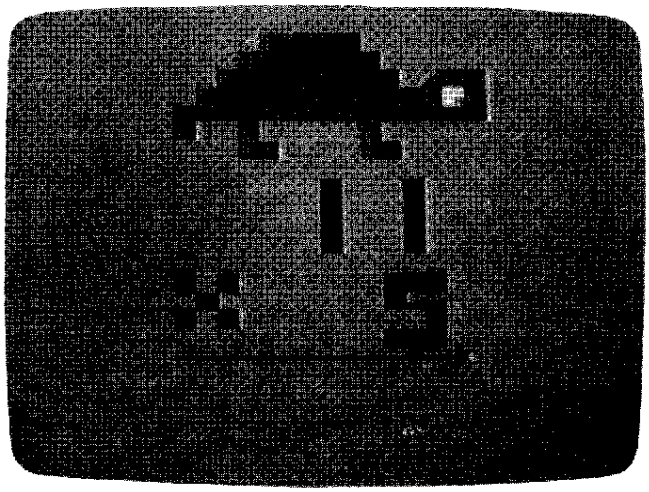
```
KK 3025 RETURN
NA 3030 P=VAL(F$(1,1))
BN 3035 X=34:GOSUB 480
ND 3040 P=VAL(F$(2,2))
BL 3045 X=40:GOSUB 480
KI 3050 RETURN
JD 6000 PLOT 27,24:DRAWTO 27,26:PLOT 2
        6,25:DRAWTO 28,25:RETURN
CE 6002 PLOT 26,25:DRAWTO 28,25:RETURN

PD 6004 PLOT 26,24:PLOT 28,24:PLOT 27,
        25:PLOT 26,26:PLOT 28,26:RETUR
        N
JO 6500 SOUND 2,150,10,10:FOR I=1 TO 5
        0:NEXT I:SOUND 2,125,10,12:FOR
         I=1 TO 50:NEXT I:SOUND 2,0,0,
        0:RETURN
DJ 6510 REM SOUND
GP 8000 DATA 28,34,40
```

Program 4: Snertle For TI-99/4A

```
100 GOTO 150
110 FOR I=1 TO LEN(H$)
120 CALL HCHAR(ROW,COL+I,ASC(SEG$(H
    $,I,1)))
130 NEXT I
140 RETURN
150 GOSUB 2710
160 CALL CLEAR
170 CALL SCREEN(12)
180 PRINT TAB(5);"** S N E R T L E
    **":::::
190 PRINT "SELECT ONE:"::
200 PRINT TAB(3);"1) ADDITION"::
210 PRINT TAB(3);"2) SUBTRACTION"::
220 PRINT TAB(3);"3) MULTIPLICATION
    "::
230 PRINT TAB(3);"4) END PROGRAM"::
    ::::
240 PRINT "(ENTER 1, 2, 3, OR 4)";
250 CALL KEY(0,Q,ST)
260 IF ST=0 THEN 250
270 Q=Q-48
280 IF (Q>4)+(Q<1)THEN 250
290 KOL=Q
300 IF Q<>2 THEN 320
310 KOL=10
320 CALL COLOR(11,KOL+4,1)
330 C=14
340 IF (Q<>1)*(Q<>2)THEN 360
350 C=99
360 IF Q=3 THEN 2210
370 IF Q=4 THEN 3100
380 CALL CLEAR
390 CALL SCREEN(4)
400 PRINT TAB(4);"ENTER LARGEST VAL
    UE:"::
410 PRINT " (LOWEST :1  HIGHEST:";C
    ;")"::
420 INPUT R
430 IF (R<1)+(R>C)THEN 420
440 PRINT ::
450 PRINT TAB(4);"ENTER SMALLEST VA
    LUE"::
460 PRINT " (LOWEST :0  HIGHEST:";R
    ;")"::
470 INPUT S
480 IF (S<0)+(S>R)THEN 470

490 CALL CLEAR
500 CALL SCREEN(10)
510 PRINT "PRESS 'X' TO RETURN TO M
    ENU":::::::::::
520 FOR I=1 TO 400
530 NEXT I
540 CALL CLEAR
550 CALL SCREEN(12)
560 Z=0
570 ZZ=0
580 GOSUB 2410
590 GOSUB 2510
600 GOSUB 2580
610 TR=0
620 ZZ=ZZ+1
630 RANDOMIZE
640 L=INT(RND*(R-S+1))+S
650 IF (Q=3)*(T=1)THEN 670
660 K=INT(RND*(R-S+1))+S
670 F$=STR$(K)
680 Y=9
690 W=15
700 IF K>=L THEN 720
710 Y=14
720 GOSUB 2840
730 Y=14
740 IF L<=K THEN 760
750 Y=9
760 F$=STR$(L)
770 GOSUB 2840
780 ON Q GOSUB 2960,2960,3040
790 IF Q<>1 THEN 810
800 M=K+L
810 IF (Q<>2)+(K<L)THEN 830
820 M=K-L
830 IF (Q<>2)+(K>=L)THEN 850
840 M=L-K
850 IF Q<>3 THEN 870
860 M=K*L
870 CALL HCHAR(18,9,104,14)
880 MM=1
890 IF M<=9 THEN 910
900 MM=2
910 IF M<=99 THEN 930
920 MM=3
930 V=0
940 GOSUB 2410
950 FOR J=0 TO MM-1
960 CALL HCHAR(22,20-4*J,94)
970 CALL KEY(0,K1,ST)
980 IF ST=0 THEN 970
```



"Snertle," TI version.

```
990 IF ((K1<48)+(K1>57))*(K1<>88)TH
    EN 970
1000 IF (K1=88)*(ZZ=1)THEN 460
1010 IF K1<>88 THEN 1060
1020 CALL CLEAR
1030 PRINT TAB(3);"PERCENTAGE :";IN
     T(Z/(ZZ-1)*100)
1040 PRINT :::
1050 GOTO 190
1060 CALL HCHAR(22,20-4*J,32)
1070 P=K1-48
1080 V=V+(P*10^J)
1090 X=19-4*J
1100 Y=20
1110 GOSUB 1430
1120 NEXT J
1130 IF M=V THEN 1310
1140 CALL SOUND(300,110,2)
1150 FOR I=20 TO 24
1160 CALL HCHAR(I,1,32,30)
1170 NEXT I
1180 IF TR=1 THEN 1230
1190 TR=1
1200 GOSUB 2660
1210 GOSUB 2010
1220 GOTO 930
1230 M$=STR$(M)
1240 FOR OO=1 TO MM
1250 P=VAL(SEG$(M$,OO,1))
1260 X=19-(MM-OO)*4
1270 GOSUB 1430
1280 NEXT OO
1290 FOR T=1 TO 400
1300 NEXT T
1310 GOSUB 2510
1320 IF TR<>0 THEN 1390
1330 CALL HCHAR(5,23,136)
1340 GOSUB 1950
1350 Z=Z+1
1360 CALL SOUND(200,196,2)
1370 CALL SOUND(200,262,2)
1380 CALL SOUND(200,294,2)
1390 FOR I=9 TO 24
1400 CALL HCHAR(I,2,32,30)
1410 NEXT I
1420 GOTO 610
1430 IF P<>0 THEN 1460
1440 GOSUB 1920
1450 RETURN
1460 ON P GOSUB 1480,1500,1550,1600
     ,1650,1710,1790,1850,1890
1470 RETURN
1480 CALL VCHAR(Y,X+1,115,4)
1490 RETURN
1500 GOSUB 2190
1510 GOSUB 2160
1520 CALL HCHAR(Y+2,X,115)
1530 GOSUB 2140
1540 RETURN
1550 GOSUB 2190
1560 GOSUB 2160
1570 CALL HCHAR(Y+2,X+2,115)
1580 GOSUB 2140
1590 RETURN
1600 CALL VCHAR(Y,X,115,2)
1610 CALL HCHAR(Y+2,X,115,3)
1620 CALL HCHAR(Y+1,X+1,114)
1630 CALL HCHAR(Y+3,X+1,114)
1640 RETURN
1650 GOSUB 2190
1660 CALL HCHAR(Y+1,X,115)
1670 CALL HCHAR(Y+1,X+1,112,2)
1680 CALL HCHAR(Y+2,X+2,115)
1690 GOSUB 2140
1700 RETURN
1710 GOSUB 2190
1720 CALL HCHAR(Y+2,X+2,115)
1730 CALL HCHAR(Y+1,X,115)
1740 CALL HCHAR(Y+1,X+1,112,2)
1750 CALL HCHAR(Y+2,X,115)
1760 CALL HCHAR(Y+2,X+2,115)
1770 GOSUB 2140
1780 RETURN
1790 GOSUB 2190
1800 CALL HCHAR(Y+1,X+2,115)
1810 CALL HCHAR(Y+2,X+1,115)
1820 CALL HCHAR(Y+2,X+2,113)
1830 CALL HCHAR(Y+3,X+1,115)
1840 RETURN
1850 GOSUB 1500
1860 CALL HCHAR(Y+1,X,115)
1870 CALL HCHAR(Y+2,X+2,115)
1880 RETURN

1890 GOSUB 1850
1900 CALL HCHAR(Y+2,X,32)
1910 RETURN
1920 GOSUB 1850
1930 CALL HCHAR(Y+1,X+1,32)
1940 RETURN
1950 H$="GOOD!"
1960 ROW=3
1970 COL=12
1980 GOSUB 110
1990 RETURN
2000 REM  CORRECT
2010 H$="TRY"
2020 ROW=2
2030 COL=13
2040 GOSUB 110
2050 H$="AGAIN"
2060 ROW=3
2070 COL=12
2080 GOSUB 110
2090 FOR I=1 TO 200
2100 NEXT I
2110 RETURN
2120 CALL VCHAR(Y,X,115,4)
2130 RETURN
2140 CALL HCHAR(Y+3,X,115,3)
2150 RETURN
2160 CALL HCHAR(Y+1,X,112,2)
2170 CALL HCHAR(Y+1,X+2,115)
2180 RETURN
2190 CALL HCHAR(Y,X,115,3)
2200 RETURN
2210 CALL CLEAR
2220 CALL SCREEN(4)
2230 PRINT "DO YOU WISH TO PRACTICE
     :":::
2240 PRINT TAB(3);"1) TIMES TABLES,
      OR"::
2250 PRINT TAB(3);"2) RANDOM NUMBER
     S ?":::::::::
2260 PRINT TAB(5);"(ENTER 1 OR 2)"
2270 CALL KEY(0,K1,ST)
2280 IF ST=0 THEN 2270
2290 IF (K1<>49)*(K1<>50)THEN 2270
2300 T=K1-48
2310 IF T=2 THEN 380
2320 CALL CLEAR
2330 PRINT TAB(6);"ENTER TIMES TABL
     E"::
```

```
2340 PRINT TAB(6);"(ENTER 1 TO 14)"
     ::
2350 INPUT K
2360 IF (K<1)+(K>14)THEN 2350
2370 S=0
2380 R=14
2390 GOTO 490
2400 REM  DRAW THE SHELL
2410 R5=5
2420 COL=13
2430 FOR I=1 TO 4
2440 CALL HCHAR(I,COL,96,R5)
2450 R5=R5+2
2460 COL=COL-1
2470 NEXT I
2480 CALL HCHAR(5,9,96,12)
2490 RETURN
2500 REM  DRAW THE HEAD
2510 CALL HCHAR(3,21,97)
2520 CALL HCHAR(3,22,96,2)
2530 CALL HCHAR(4,21,96,3)
2540 CALL HCHAR(4,22,128)
2550 CALL HCHAR(5,21,96,3)
2560 RETURN
2570 REM  DRAW THE FEET AND TAIL
2580 FOR I=1 TO 8
2590 READ R5,C
2600 CALL HCHAR(R5,C,96)
2610 NEXT I
2620 RESTORE
2630 DATA 6,9,6,12,6,18,7,12,7,13,7
     ,18,7,19,5,22
2640 RETURN
2650 REM  ERASE THE HEAD
2660 FOR I=3 TO 5
2670 CALL HCHAR(I,21,32,3)
2680 NEXT I
2690 RETURN
2700 REM  DEFINE CHARS & COLORS
2710 CALL CHAR(96,"FFFFFFFFFFFFFFFF
     ")
2720 CALL CHAR(97,"0103070F1F3F7FFF
     ")
2730 CALL CHAR(104,"000000FFFF00000
     0")
2740 CALL CHAR(128,"00000000F0F0F0F
     F")
2750 CALL CHAR(136,"303018ØC07Ø3ØØØ
     Ø")
2760 CALL COLOR(9,3,1)
2770 CALL COLOR(13,6,16)
2780 CALL COLOR(14,14,3)
2790 CALL CHAR(112,"00000000FFFFFFF
     F")
2800 CALL CHAR(113,"F0F0F0F0F0F0F0F
     0")
2810 CALL CHAR(114,"0707070707070707
     ")
2820 CALL CHAR(115,"FFFFFFFFFFFFFFF
     F")
2830 RETURN
2840 IF LEN(F$)=2 THEN 2890
2850 P=VAL(SEG$(F$,1,1))
2860 X=W+4
2870 GOSUB 1430
2880 RETURN
2890 P=VAL(SEG$(F$,1,1))
2900 X=W
2910 GOSUB 1430
2920 P=VAL(SEG$(F$,2,1))
2930 X=W+4
2940 GOSUB 1430
2950 RETURN
2960 CALL VCHAR(14,11,115,3)
2970 CALL HCHAR(15,10,115)
2980 CALL HCHAR(15,12,115)
2990 IF Q=2 THEN 3010
3000 RETURN
3010 CALL HCHAR(14,11,32)
3020 CALL HCHAR(16,11,32)
3030 RETURN
3040 CALL HCHAR(14,9,115)
3050 CALL HCHAR(14,11,115)
3060 CALL HCHAR(15,10,115)
3070 CALL HCHAR(16,9,115)
3080 CALL HCHAR(16,11,115)
3090 RETURN
3100 END
```

Program 5: Snertle For The Color Computer

```
100 CLS(1):B$=CHR$(32)
110 PRINT@74,"**SNERTLE**"
120 PRINT@138,"SELECT 1"
130 PRINT@202,"1) ADDITION"
140 PRINTTAB(10)"2) SUBTRACTION"
150 PRINTTAB(10)"3) MULTIPLICATION"
155 PRINTTAB(10)"4) END"
160 PRINTTAB(10)"(ENTER 1,2,3 OR 4)
    ";:INPUTQ:IF Q>4 OR Q<1 THEN 16
    0
185 C=14:IF Q=1 OR Q=2 THEN C=99
187 IF Q=3 THEN 1000
188 IF Q=4 THEN END
190 CLS(1):PRINT@37,"ENTER LARGEST
    VALUE"
200 PRINTTAB(5)"(MIN.:1  MAX.:";C;"
    )";:INPUTR:IF R<1 OR R>C THEN 2
    00
230 PRINT@133,"ENTER SMALLEST VALUE
    "
240 PRINTTAB(5)"(MIN.:0  MAX.:";R;"
    )";:INPUTS:IF S<0 OR S>R THEN 2
    40
263 CLS:PRINT@227,"PRESS ⊠ TO RETUR
    N TO MENU";:FORI=1TO750:NEXTI:C
    LS(0)
270 Z=0:ZZ=0
275 GOSUB 1100:GOSUB 1170:GOSUB1230
301 TR=0:ZZ=ZZ+1
305 L=INT(RND(R-S)+S)
310 IF Q=3ANDT=1THEN320
315 K=INT(RND(R-S)+S)
320 F$=STR$(K):W=0
325 IF K<L AND Q=2  THEN TR=0:GOTO3
    05
330 W=0:GOSUB3000
335 W=64
340 F$=STR$(L)
345 W=96:GOSUB 3000
346 ON Q GOSUB 6000,6000,6004
350 IF Q=1 THEN M=K+L
355 IF Q=2 THEN M=K-L
360 IF Q=3 THEN M=K*L
380 MM=1:IF M>9 THEN MM=2
385 IF M>99 THEN MM=3
390 GOSUB 740
393 V=0:GOSUB 1100
395 FOR J=0 TO MM-1
397 POKE 1466-(4*J),94
399 HH$=INKEY$
400 H$=INKEY$
405 IF H$=""THEN 400
410 IF H$="X" AND ZZ=1 THEN 100
```

PENTOMINOS
A Puzzle-Solving Program

Jim Butterfield, Associate Editor

Computers can solve puzzles. With the right set of instructions, a program will follow the same logic as humans, trying things to see if they fit. It's interesting to watch the computer working in this way.

This famous puzzle is dealt with at some length in Arthur C. Clarke's novel Imperial Earth. *The characters of the novel don't use a computer to solve the puzzle.*

The original program works on all Commodore computers. Additional versions are included here for the Atari, IBM PC and PCjr, TI-99/4A, Radio Shack Color Computer, and Apple.

NOTE: *IBM, TI, Color Computer, and Apple users should insert lines 110–860 from Program 1, the Commodore version, into their programs. The* rem *statements at the ends of these lines should be ignored.*

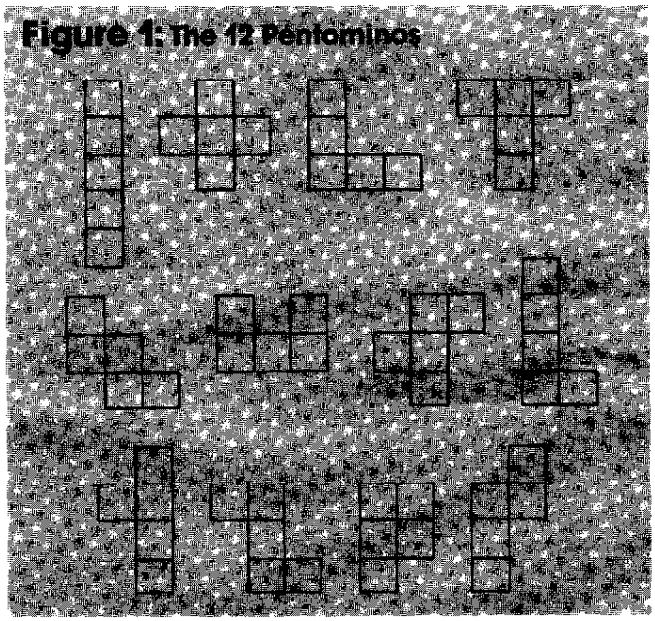
Pentominos are like dominos, except that they are made up of five elements rather than two. If we put five squares end to end and glued them together, we'd get a long strip, often called the I pentomino. On the other hand, if we took a central square and glued the other four squares to the sides, top, and bottom, we'd get something that looks like a plus sign, which many people call the X pentomino.

Allowing for the differences that are caused by rotating or turning over a piece, there are 12 different pentominos. They are shown in Figure 1; but you might find it fun to try discovering them yourself by drawing them out on a piece of paper. Most of them look a little like letters—you can see a T, an X, and a W among them, for example.

What's The Puzzle?

The 12 different pentominos, each with an area of 5 squares, give a total of 60 squares. Suppose you had to cut these pentominos out of a rectangle without wasting any space: How big would the rectangle need to be?

Figure 1: The 12 Pentominos



We know two things: The total area is 60 squares; and the rectangle must be at least three wide (otherwise, we couldn't cut out the plus sign). So it might be possible to get all the pentominos from a rectangle that is 3 x 20, or 4 x 15, or 5 x 12, or 6 x 10. As it turns out, we can do it in any of these ways.

We can turn the question inside out and put it this way: Can you fit all 12 pentominos into a rectangle of size: 3 x 20, or 4 x 15, or 5 x 12, or 6 x 10?

The Brain Bender

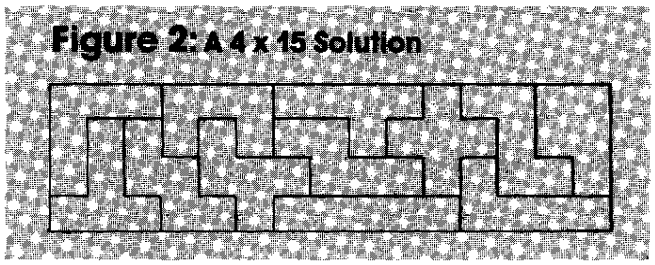
Don't let the following computer program take the fun out of the puzzle for you. Cut the pieces out of cardboard and try your hand at the puzzle.

```
265 CLS
270 Z=0:ZZ=0
275 COLOR 2:GOSUB 1100:GOSUB 1170:GOSUB
1230:GOSUB 1260: COLOR Q *2
301 TR=0:ZZ=ZZ+1
305 L=INT(RND(1)*(R-S+1))+S
310 IF Q=3 AND T=1 THEN 320
315 K=INT(RND(1)*(R-S+1))+S
320 F$=STR$(K):W=0
325 IF K<L THEN W=5
330 GOSUB 3000
335 W=5
337 IF L>K THEN W=0
340 F$= STR$(L)
345 GOSUB 3000
346 ON Q GOSUB 6000,6000,6004
    IF Q=1 THEN M=K+L
350   Q=2 AND K>=L THEN M=K-L
360 IF Q=2 AND K<L THEN M=L-K
365 IF Q=3 THEN M=K*L
380 GOSUB 740:MM=1:IF M>9 THEN MM=2
385 IF M>99 THEN MM=3
390 GOSUB 740
393 V=0:COLOR 2 :GOSUB 1100:COLOR Q*2
394 FOR A=1 TO 10:B$=INKEY$:NEXT
395 FOR J=0 TO (MM-1)
397 LOCATE 24,30-4*J:PRINT"^";
400 H$=INKEY$
405 IF H$="X"AND ZZ=1 THEN 100
406 IF H$="X" THEN CLS:PRINT B$"PERCENTA
GE:";INT(Z/(ZZ-1)*100):GOTO 120
407 IF H$="" OR H$<"0" OR H$>"9" THEN 40
0
412 FOR I= 21 TO 31:LOCATE 24,I:PRINT SP
$;:NEXT
415 P=VAL (H$):Y=20
420 V=V+(P*10^J):X=29-J*4:GOSUB 475:NEXT
 J
450 IF M=V THEN 470
452 FOR I= 20 TO 23:LOCATE I,21:FOR J=1
TO 11:PRINT SP$;:NEXT J,I
456 IF TR =1 THEN 460
458 TR =1:GOSUB 1500:GOSUB 770:GOTO 395
460 M$ =STR$(M):X =33:Y=20
462 FOR OO=MM TO 1 STEP -1
464 P = VAL (MID$ (M$,(OO+1),1))
465 X=X-4:GOSUB 475:NEXT OO:RESTORE
470 FOR I=1 TO 750:NEXT:GOSUB 1230: IF T
R=0 THEN GOSUB 2500::GOSUB 755: Z=Z+1:GO
SUB 6500
471 GOSUB 2225: GOTO 301
475 LOCATE Y,X
480 IF P=0 THEN GOSUB 720
485 ON P GOSUB 500,525,555,585,610,633,6
60,680,700:RETURN
500 PRINT R$R$;:FOR I=1 TO 4 :PRINT S$D$
$;:NEXT :RETURN
525 PRINT S$S$S$D$L$S$D$L$TB$L$L$TB$L$L$
BD$L$S$S$S$:RETURN
555 PRINT S$S$S$D$L$S$D$L$S$L$L$TB$D$L$L
$S$S$:RETURN
585 PRINT LB$R$S$D$L$L$L$S$S$S$D$L$S$D$L
$:RETURN
610 PRINT S$S$S$D$L$L$L$S$BB$BB$D$L$S$D$
L$L$L$S$S$S$:RETURN
633 PRINT S$S$S$D$L$L$L$S$BB$BB$D$L$L$L$
S$R$S$D$L$L$L$S$S$S$:RETURN
660 PRINT S$S$S$D$L$S$D$L$L$S$D$L$L$S$:R
ETURN
680 PRINT S$S$S$D$L$L$L$S$BB$S$D$L$L$L$S
$R$S$D$L$L$L$S$S$S$S$:RETURN
700 PRINT S$S$S$D$L$L$L$S$BB$S$D$L$S$D$L
$S$:RETURN
720 PRINT S$S$S$D$L$L$L$S$R$S$D$L$L$L$S$
R$S$D$L$L$L$S$S$S$S$:RETURN
740 LOCATE 18,21:FOR I=1 TO 11:PRINT BB$
;:NEXT:RETURN
755 LOCATE 4,7:PRINT "GOOD!":RETURN
770 LOCATE 3,8:PRINT "TRY" D$L$L$L$L$ "A
GAIN"
780 FOR I=1000 TO 500 STEP -250:SOUND I,
4:NEXT:FOR TD=1 TO 500:NEXT:RETURN
960 FOR I=1 TO 4:LOCATE X,I:PRINT S$:NEX
T:RETURN
1000 CLS:LOCATE 7,10:PRINT"DO YOU WISH T
O:"
1010 PRINT:PRINT:PRINT C$"1) PRACTICE TI
MES TABLE"
1020 PRINT:PRINT C$"2) RANDOM NUMBERS
1030 PRINT:PRINT:PRINT C$"(ENTER 1 OR 2)
";:INPUT T:IF T<1 OR T>2 THEN PRINT U$U$
U$U$:GOTO 1030
1050 IF T=2 THEN GOTO 190
1060 CLS:PRINT:PRINT:PRINT C$"ENTER TIME
S TABLE"
1070 PRINT:PRINT C$"(1-14)";:INPUT K:IF
K<1 OR K>14 THEN PRINT U$U$U$:GOTO 1070
1090 S=0:R=14:GOTO 263
1100 FOR I= 2 TO 6
1110 READ A :READ B
1120 FOR J= 1 TO B
1130 LOCATE I,J+A :PRINT CHR$(176)
1140 NEXT J:NEXT I:RESTORE:RETURN
1170 LOCATE 7,4:FOR I= 1 TO 11 :PRINT TB
$;:NEXT :RETURN
1230 COLOR 2:LOCATE 5,15:PRINT CHR$(47)U
$BB$BB$D$L$CHR$(249)LB$D$L$LB$D$L$L$L$L$
TB$TB$TB$:COLOR Q*2:RETURN
1240 LOCATE 7,5:PRINT S$ :LOCATE 7,14:PR
INT S$
1250 RETURN
1260 COLOR 2:GOSUB 1240:LOCATE 9,5,PRINT
 TB$TB$:LOCATE 8,14:PRINT TB$TB$:RETURN:
COLOR Q *2
1270 RETURN
1500 FOR I=4 TO 7:LOCATE I,15:FOR J=1 TO
 4:PRINT SP$;:NEXT J,I:RETURN
2225 FOR I= 9 TO 23:LOCATE I,21: FOR J=
1 TO 11 :PRINT SP$;:NEXT J,I:RETURN
2500 COLOR 2:LOCATE 6,17:PRINT CHR$(126)
:RETURN:COLOR Q*2
3000 COLOR Q*2:X=29:IF LEN (F$)>2 THEN 3
030
3015 F=VAL (MID$(F$,2,1))
3020 Y=9+W:GOSUB 475
3025 RETURN
3030 P=VAL(MID$(F$,3,1))
3035 Y=9+W:GOSUB 475
3040 P=VAL(MID$(F$,2,1))
3045 X=X-4:GOSUB 475
3050 RETURN
5000 DATA 6,5,5,7,4,9,3,11,3,11
6000 LOCATE 14,22:PRINT S$D$L$L$S$S$S$D$
L$L$S$;
6002 IF Q=2 THEN PRINT L$SP$U$U$L$SP$
6003 RETURN
6004 LOCATE 14,21:PRINT S$D$S$U$S$D$D$L$
L$L$S$R$S$:RETURN
6500 FOR I=500 TO 1000 STEP 250:SOUND I,
4:NEXT:RETURN
```

It's an interesting way to wile away the hours. 6 x 10 and 5 x 12 are not too hard; 4 x 15 will make you work; and 3 x 20, which seems at first to be the easiest, proves to be a real brain bender.

A sample solution to the 4 x 15 problem is given in Figure 2.

Figure 2: A 4 x 15 Solution



If humans can waste time trying to fit the pieces, computers can do it too. "Pentominos" does not run at blinding speed; it tries the pieces at about the same speed as humans do. It's dumber than human puzzle solvers. It will try to make a piece fit in places we know instinctively are hopeless. But the computer has no intuition: It will plod along, making dumb moves until it finds a combination that fits.

The program tries the pieces "visibly"—that is, you can see it putting the pieces in place, thinking about its next move, and then taking a piece back out when it becomes obvious (even to the dumb computer) that it can't work there.

In a moment we'll get to more detail on how it works. The computer always thinks about fitting the upper-leftmost empty square, and it will tell you which piece it is trying to fit there; that piece's identity will be shown in a corner of the screen. So you can track the computer's thoughts if you wish.

It can take a few minutes or several hours to find the next solution. This program is a good one to set up for an overnight run. You might want to turn off your TV set or monitor and let the computer hum away quietly all by itself.

When a solution is found, you can type CONT at any blank place on the screen, and the computer will go after the next solution.

How It Works

The pentominos and all their possible rotations are stored in DATA statements. Only four squares need to be described for each pentomino rotation, since the information gives coordinates based upon the starting square.

After reading in the data, the computer uses the following logic. Line numbers are given for those who would like to try examining the program.

1. (Line 2010) The computer looks through the list of pieces to find the first one that isn't being used. Then it searches the board for a blank square, starting at the left and searching each column top to bottom. That's the next place it will try to fit a piece. If it can't find a blank, we have a solution and will go to step 5.

2. (Line 2030) The piece just picked is set to its first rotation.

3. (Line 2060) The computer tries to fit the piece starting at the square it has identified. If it doesn't fit, it will skip ahead to step 7.

4. (Line 2120) The piece fits, so the computer puts it onto the board, onto the screen, and marks off the piece as used. It then goes back to step 1 to look for a new place to fit pieces.

5. (Line 2170) We have a solution! Stop and wait for the user to admire us. If the user types CONT, we'll keep going into step 6.

6. (Line 2190) We've reached a dead end, so we go back and remove the last piece placed on the board. If there are no pieces left, we quit; at this point we will have found all the solutions.

7. (Line 2260) Let's rotate the current piece so that we can try it in a different way. If we can find a new rotation, we go back to step 3 to try the piece. If not, we continue to step 8.

8. (Line 2300) The computer looks through the list of pieces to find the next piece to be tried. Then it goes back to step 2.

Variables And Arrays

If you're trying to read the program, it will be worthwhile to have some information on variables and arrays. Here are some useful ones:

Array B(X,Y) is the board. If the value is zero, that part of the board is blank. When a board square is used, the appropriate value in this array is set to the number of the occupying piece; but the important thing to remember is that it's set to nonzero.

The DATA statements show all rotations of all pieces. They are transferred to arrays X and Y:

Arrays X(rotation,C) and Y(rotation,C) tell where to find the squares (X and Y) of each piece's rotation. The rotation is taken from the DATA statements.

Array P(rotation) tells which piece is involved for each rotation of the above table.

Each Piece Has Data

Array P$(piece) is the name of the piece.

Array S(piece) tells where to find the starting rotation for piece X.

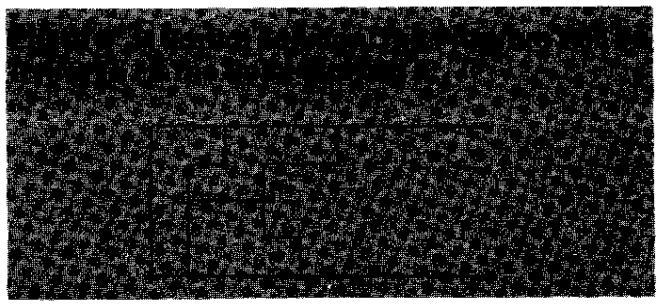
Array T(piece) tells which rotation is currentl being used (or tried) for piece X.

Arrays X2(piece) and X2(piece) list the startin square where piece A has been placed.

Tracking The Moves

Array U(move) lists the pieces in the order in which we tried them.

The piece under consideration is designated



by P; its current rotation, of course, will be T(P).

When we place a piece, we log it into array U and use P1 to keep track of how many pieces have been used.

Program Variations

The program could be speeded up significantly by using a compiler or by converting it to machine language. I have chosen not to do that for two reasons: compatibility and readability.

A machine language version would nevertheless be quite straightforward to write. No special math or other logic is involved. Such a program would be very fast. But it would not be universal, since different machines would need to load the program into different memory locations.

If you go for many solutions, you should realize that some of the solutions are transformations of others. Given one solution, others can be found by inverting it left to right or top to bottom. This means that each solution is really four solutions; but the computer will find each of the four as it works. If this is not desired, the extra solutions can be eliminated by removing all but two of the rotations of a single eight-rotation piece. That way, the reflected solutions couldn't happen: That piece can appear in only one orientation.

For example, we could eliminate reflected solutions by changing line 770 to DATA R,2 and then deleting lines 800 to 850 inclusive.

Making It Smarter

The program would run faster if it didn't show its moves on the screen, but watching it work is most of the fun. For one thing, it may remind you of an important aspect of computers: They're dumb, but they're faithful.

The computer will lumber along, trying dumb moves. But it won't get tired, and it will eventually reach the solution.

Yes, we could add extra logic to make the computer smarter. We could ask the computer to scan for some of the obviously impossible situations that it does not recognize at all with the present program. But there's a danger: The computer could waste more time being smart than it does being dumb.

Program 1: Pentominos For Commodore

Refer to the "Automatic Proofreader" article before typing this program in.

```
100 PRINT CHR$(142)"{CLR}{5 RIGHT}PENTOMI
    NOS{DOWN}"                          :rem 140
110 DATA I,2                            :rem 83
120 DATA 0,1,0,2,0,3,0,4                :rem 107
130 DATA 1,0,2,0,3,0,4,0                :rem 108
140 DATA X,1                            :rem 100
150 DATA 1,-1,1,0,2,0,1,1               :rem 152
160 DATA V,4                            :rem 103
170 DATA 0,1,0,2,1,0,2,0                :rem 108
180 DATA 0,1,0,2,1,2,2,2                :rem 113
190 DATA 1,0,2,0,2,1,2,2                :rem 114
200 DATA 1.0.2.0.2.-1.2.-2              :rem 196
210 DATA T,4                             :rem 97
220 DATA 0,1,0,2,1,1,2,1                :rem 106
230 DATA 1,0,1,1,2,0,1,2                :rem 107
240 DATA 1,0,2,0,1,-1,1,-2              :rem 198
250 DATA 2,-1,2,0,2,1,1,0               :rem 155
260 DATA W,4                            :rem 105
270 DATA 0,1,1,1,1,2,2,2                :rem 113
280 DATA 1,0,1,1,2,1,2,2                :rem 114
290 DATA 0,1,1,-1,1,0,2,-1              :rem 202
300 DATA 1,-1,1,0,2,-2,2,-1             :rem 242
310 DATA U,4                             :rem 99
320 DATA 0,2,1,0,1,1,1,2                :rem 107
330 DATA 2,0,0,1,1,1,2,1                :rem 108
340 DATA 0,1,1,0,2,0,2,1                :rem 108
350 DATA 1,0,0,1,0,2,1,2                :rem 109
360 DATA F,8                             :rem 93
370 DATA 0,1,1,-1,1,0,2,0               :rem 155
380 DATA 1,-1,2,-1,1,0,1,1              :rem 203
390 DATA 1,-1,1,0,1,1,2,1               :rem 159
400 DATA 1,-1,1,0,2,0,2,1               :rem 151
410 DATA 0,1,1,1,1,2,2,1                :rem 108
420 DATA 1,0,1,1,2,1,1,2                :rem 109
430 DATA 1,0,1,1,2,-1,2,0               :rem 154
440 DATA 1,-2,1,-1,2,-1,1,0             :rem 246
450 DATA L,8                             :rem 99
460 DATA 1,0,2,0,3,0,3,1                :rem 114
470 DATA 0,1,0,2,0,3,1,3                :rem 115
480 DATA 1,-3,1,-2,1,-1,1,0             :rem 251
490 DATA 1,0,2,0,3,0,3,-1               :rem 162
500 DATA 1,0,2,0,3,0,0,1                :rem 106
510 DATA 0,1,0,2,0,3,1,0                :rem 107
520 DATA 0,1,1,1,2,1,3,1                :rem 111
530 DATA 1,0,1,1,1,2,1,3                :rem 112
540 DATA Y,8                            :rem 112
550 DATA 0,1,0,2,0,3,1,1                :rem 112
560 DATA 1,0,2,0,3,0,1,1                :rem 113
570 DATA 1,-1,1,0,1,1,1,2               :rem 159
580 DATA 1,-1,1,0,2,0,3,0               :rem 161
590 DATA 0,1,0,2,0,3,1,2                :rem 11
600 DATA 1,0,2,0,3,0,2,1                :rem 10
610 DATA 1,-2,1,-1,1,0,1,1              :rem 19
620 DATA 1,0,2,0,3,0,2,-1               :rem 15
630 DATA Z,4                            :rem 109
640 DATA 0,1,1,1,2,1,2,2                :rem 114
650 DATA 1,0,1,1,1,2,2,2                :rem 115
660 DATA 1,-2,1,-1,1,0,2,-2             :rem 251
670 DATA 2,-1,1,0,2,0,0,1               :rem 159
680 DATA P,8                            :rem 108
690 DATA 0,1,1,0,1,1,2,0                :rem 115
700 DATA 1,0,0,1,1,1,0,2                :rem 107
710 DATA 0,1,1,0,1,1,1,2                :rem 109
720 DATA 1,0,0,1,1,1,2,1                :rem 110
730 DATA 1,-1,1,0,2,-1,2,0              :rem 202
740 DATA 1,-1,1,0,0,1,1,1               :rem 156
750 DATA 0,1,0,2,1,1,1,2                :rem 114
```

```
2070 X=X(T(P),J-1)+X1:Y=Y(T(P),J-1)+Y1:X
1(J)=X:Y1(J)=Y
2080 IF X<1 OR Y<1 OR X>W2 OR Y>W1 GOTO
2260
2090 IF B(Y,X)<>0 GOTO 2260
2100 NEXT J
2110 REM IT FITS - PUT PIECE IN PLACE
2120 B=P:FOR J=0 TO 4
2130 X=X1(J):Y=Y1(J):GOSUB 3500
2140 NEXT J
2150 X2(P)=X1:Y2(P)=Y1:P1=P1+1:U(P1)=P:G
OTO 2010
2160 REM BOARD FILLED
2170 LOCATE 15,1:PRINT "  SOLUTION";:END
2180 REM UNDRAW LAST ONE
2190 P=U(P1):U(P1)=0:P1=P1-1:IF P1<0 THE
N PRINT"THAT'S ALL":END
2200 B=0:X=X2(P):Y=Y2(P):C$=" ":GOSUB 35
00
2210 X1=X:Y1=Y:FOR J=1 TO 4
2220 X=X(T(P),J-1)+X1:Y=Y(T(P),J-1)+Y1:X
1(J)=X:Y1(J)=Y
2230 GOSUB 3500
2240 NEXT J
2250 REM ROTATE THE PIECE
2260 T(P)=T(P)+1:IF P(T(P))=P GOTO 2060
2270 REM GIVE UP ON PIECE
2280 T(P)=0
2290 REM LOOK FOR NEW PIECE
2300 P=P+1:IF P>12 GOTO 2190
2310 IF T(P)<>0 GOTO 2300
2320 GOTO 2030
3000 FOR J=1 TO 12:IF T(J)<>0 THEN NEXT
J
3010 RETURN
3200 FOR X1=1 TO W2:FOR Y1=1 TO W1
3210 IF B(Y1,X1)=0 GOTO 3230
3220 NEXT Y1,X1
3230 RETURN
3500 LOCATE Y+2,X:PRINT C$:B(Y,X)=B
3510 RETURN
```

Program 4: Pentominos For TI-99/4A

Insert lines 110–860 from the Commodore version (Program 1).

*(**Note:** If using a disk drive, type CALL FILES(1) before loading and running this program.)*

```
40 CALL CLEAR
50 PRINT "{8 SPACES}PENTOMINOS": :
60 GOTO 870
70 FOR I=1 TO LEN(A$)
80 CALL HCHAR(ROW,COL+I,ASC(SEG$(A$
   ,I,1)))
90 NEXT I
100 RETURN
870 DIM XX(63,4),YY(63,4),PP(64),PP
    $(13),SS(13),TT(13),BB(6,20)
880 DIM XX1(5),YY1(5),XX2(12),YY2(1
    2),UU(12)
890 CT=5
900 READ P$,N
910 IF N=0 THEN 1040
920 T=T+1
930 PP$(T)=P$
940 SS(T)=V+1
950 FOR J=V+1 TO V+N
960 PP(J)=T
970 FOR K=0 TO 3
980 READ XX(J,K),YY(J,K)
990 NEXT K
1000 NEXT J
1010 V=V+N
1020 PRINT P$;
1030 GOTO 900
1040 CALL CLEAR
1050 PRINT " CHOOSE:": :
1060 FOR J=3 TO 6
1070 PRINT J;" BY ";60/J
1080 NEXT J
1090 PRINT
1100 INPUT " SELECT 3 THRU 6: ":W1
1110 IF (W1<3)+(W1>6)+(W1<>INT(W1))
    THEN 1040
1120 W2=60/W1
1130 CALL CLEAR
1140 REM FIND NEW SPACE TO FILL
1150 GOSUB 1930
1160 P=J
1170 GOSUB 1970
1180 IF X1>W2 THEN 1500
1190 REM GET A NEW PIECE
1200 TT(P)=SS(P)
1210 ROW=CT
1220 COL=5+CT
1230 A$=PP$(P)
1240 GOSUB 70
1250 REM TRY FITTING PIECE
1260 C$=PP$(P)
1270 XX1(0)=X1
1280 YY1(0)=Y1
1290 FOR J=1 TO 4
1300 X=XX(TT(P),J-1)+X1
1310 Y=YY(TT(P),J-1)+Y1
1320 XX1(J)=X
1330 YY1(J)=Y
1340 IF (X<1)+(Y<1)+(X>W2)+(Y>W1)TH
    EN 1840
1350 IF BB(Y,X)<>0 THEN 1840
1360 NEXT J
1370 REM IT FITS - PUT PIECE IN PLA
    CE
1380 B=P
1390 FOR J=0 TO 4
1400 X=XX1(J)
1410 Y=YY1(J)
1420 GOSUB 2030
1430 NEXT J
1440 XX2(P)=X1
1450 YY2(P)=Y1
1460 P1=P1+1
1470 UU(P1)=P
1480 GOTO 1150
1490 REM BOARD FILLED
1500 ROW=15
1510 COL=5+CT
1520 A$="SOLUTION"
1530 GOSUB 70
1540 ROW=17
1550 COL=5
1560 A$="FIND ANOTHER SOLUTION?"
1570 GOSUB 70
1580 CALL KEY(3,K,S)
1590 IF S<>1 THEN 1580
1600 IF CHR$(K)="Y" THEN 1620
1610 END
1620 REM UNDRAW LAST ONE
1630 P=UU(P1)
1640 UU(P1)=0
1650 P1=P1-1
1660 IF P1>=0 THEN 1690
1670 PRINT "THAT'S ALL"
1680 STOP
1690 B=0
1700 X=XX2(P)
```

```
1710 Y=YY2(P)
1720 C$=" "
1730 GOSUB 2030
1740 X1=X
1750 Y1=Y
1760 FOR J=1 TO 4
1770 X=XX(TT(P),J-1)+X1
1780 Y=YY(TT(P),J-1)+Y1
1790 XX1(J)=X
1800 YY1(J)=Y
1810 GOSUB 2030
1820 NEXT J
1830 REM' ROTATE THE PIECE
1840 TT(P)=TT(P)+1
1850 IF PP(TT(P))=P THEN 1260
1860 REM GIVE UP ON PIECE
1870 TT(P)=0
1880 REM LOOK FOR NEW PIECE
1890 P=P+1
1900 IF P>12 THEN 1630
1910 IF TT(P)<>0 THEN 1890
1920 GOTO 1200
1930 FOR J=1 TO 12
1940 IF TT(J)=0 THEN 1960
1950 NEXT J
1960 RETURN
1970 FOR X1=1 TO W2
1980 FOR Y1=1 TO W1
1990 IF BB(Y1,X1)=0 THEN 2020
2000 NEXT Y1
2010 NEXT X1
2020 RETURN
2030 ROW=Y+1+CT
2040 COL=X+CT
2050 A$=C$
2060 GOSUB 70
2070 BB(Y,X)=B
2080 RETURN
```

Program 5:
Pentominos For The Color Computer

Insert lines 110–860 from the Commodore version (Program 1).

```
100 CLS:PRINT"{11 SPACES}PENTOMINOS"
999 PCLEAR 1
1000 DIM X(63,4),Y(63,4),P(64),P$(1
     3),S(13),T(13),B(6,20)
1001 DIM X1(5),Y1(5),X2(12),Y2(12),
     U(12)
1010 READ P$,N:IF N=0 GOTO 1070
1020 T=T+1:P$(T)=P$:S(T)=V+1
1030 FOR J=V+1 TO V+N: P(J)=T
1040 FOR K=0 TO 3:READ X(J,K),Y(J,K
     ):NEXT K,J
1050 V=V+N:PRINT P$;
1060 GOTO 1010
1070 PRINT@64,"CHOOSE:"
1080 FOR J=3 TO 6:PRINT J;" BY";60/
     J:NEXT J
1090 INPUT "SELECT 3 THRU 6";W1
1100 IF W1<3 OR W1>6 OR W1<>INT(W1)
      GOTO 1070
1110 W2=60/W1
1120 CLS
2000 REM FIND NEW SPACE TO FILL
2010 GOSUB 3000:P=J:GOSUB 3200:IF X
     1>W2 GOTO 2170
2020 REM GET A NEW PIECE
2030 T(P)=S(P)
2040 PRINT@33,P$(P)
2050 REM TRY FITTING PIECE
2060 C$=P$(P):X1(0)=X1:Y1(0)=Y1:FOR
      J=1 TO 4
2070 X=X(T(P),J-1)+X1:Y=Y(T(P),J-1)
     +Y1:X1(J)=X:Y1(J)=Y
2080 IF X<1 OR Y<1 OR X>W2 OR Y>W1
     GOTO 2260
2090 IF B(Y,X)<>0 GOTO 2260
2100 NEXT J
2110 REM IT FITS - PUT PIECE IN PLA
     CE
2120 B=P:FOR J=0 TO 4
2130 X=X1(J):Y=Y1(J):GOSUB 3500
2140 NEXT J
2150 X2(P)=X1:Y2(P)=Y1:P1=P1+1:U(P1
     )=P:GOTO 2010
2160 REM BOARD FILLED
2170 PRINT@385,"SOLUTION":END
2180 REM UNDRAW LAST ONE
2190 P=U(P1):U(P1)=0:P1=P1-1:IF P1<
     0 THEN PRINT"THAT'S ALL":END
2200 B=0:X=X2(P):Y=Y2(P):C$=" ":GOS
     UB 3500
2210 X1=X:Y1=Y:FOR J=1 TO 4
2220 X=X(T(P),J-1)+X1:Y=Y(T(P),J-1)
     +Y1:X1(J)=X:Y1(J)=Y
2230 GOSUB 3500
2240 NEXT J
2250 REM ROTATE THE PIECE
2260 T(P)=T(P)+1:IF P(T(P))=P GOTO
     2060
2270 REM GIVE UP ON PIECE
2280 T(P)=0
2290 REM LOOK FOR NEW PIECE
2300 P=P+1:IF P>12 GOTO 2190
2310 IF T(P)<>0 GOTO 2300
2320 GOTO 2030
3000 FOR J=1 TO 12:IF T(J)<>0 THEN
     NEXT J
3010 RETURN
3200 FOR X1=1 TO W2:FOR Y1=1 TO W1
3210 IF B(Y1,X1)=0 GOTO 3230
3220 NEXT Y1,X1
3230 RETURN
3500 PRINT @X+(Y+2)*32,C$;:B(Y,X)=B
3510 RETURN
```

Program 6: Pentominos For The Apple

Insert lines 110–860 from the Commodore version (Program 1).

```
1000  DIM X(63,4),Y(63,4),P(64),P$(13),
     S(13),T(13),B(6,20)
1001  DIM X1(5),Y1(5),X2(12),Y2(12),U(1
     2)
1003  HOME : HTAB 16: PRINT "PENTOMINOS
     ": PRINT
1010  READ P$,N: IF N = 0 GOTO 1070
1020 T = T + 1:P$(T) = P$:S(T) = V + 1
1030  FOR J = V + 1 TO V + N:P(J) = T
1040  FOR K = 0 TO 3: READ X(J,K),Y(J,K
     ): NEXT K,J
1050 V = V + N: PRINT P$;
1060  GOTO 1010
1070  PRINT : VTAB (5): PRINT "CHOOSE:"
     : PRINT
1080  FOR J = 3 TO 6: PRINT J;" BY ";60
      / J: PRINT : NEXT J
1090  INPUT "SELECT 3 THRU 6? ";W1
1100  IF W1 < 3 OR W1 > 6 OR W1 <  >  INT
     (W1) GOTO 1070
1110 W2 = 60 / W1
1120  HOME
2000  REM  FIND NEW SPACE TO FILL
2010  GOSUB 3000:P = J: GOSUB 3200: IF
     X1 > W2 GOTO 2170
```

PROGRAMMING THE TI

C. Regena

File Processing
Part 3

This month C. Regena concludes her three-part discussion on creating data files.

A Birthday List

Program 1 prints a birthday list of the students in a class. The same data file is used, and the information is arranged in order by birthdate. Line 180 is the OPEN statement for the printer (use your own printer configuration). Line 190 is the OPEN statement for the disk drive to read in information.

Line 210 reads in the date—again, in the same order that the items were saved. We will ignore some of the information, but all the items must be read in order. Line 250 combines several of the items into one variable T$. The birthday BD and T$ are actually arrays, so the items may be sorted. Lines 280–350 contain the sorting procedure to sort by birthday.

Line 360 and lines 510–560 print the header. Lines 370–480 print the information. Lines 380–400 print the month and day from the BD number that was saved. Line 410 prints a blank line between months. Lines 420–450 use POS and SEG$ to separate the T$ item back into its parts, then line 460 prints the information in columns using the IMAGE statement of line 200.

The Report Writer

Program 2 generates reports using the data saved in Program 1 of Part 2 (April 1984). Lines 160–200 present the option to print the report for one of the reading groups or for the whole class.

These reports will use a 132-column line, or *compressed print* (16.5 characters per inch). Line 210 OPENs device #1 for the printer. The previous reports used an 80 column line, which is the default value for most printers. VARIABLE 132 is used to designate a longer line before a carriage return. Line 230 sets *my* printer (TI 825, which is like the TI 840) to use compressed print. You will probably need a different command.

Some printers can use a certain CHR$ number. Other printers may require you to set certain hardware switches. I have used compressed print and the 132-column line so more can fit on the one line. The other two reports in this program may be printed with the regular printing.

Line 240 is the OPEN statement to read the data from the data file created by Program 1 (Part 2, April). Again, the variables are in the same order as they were saved. Line 280 checks for the end of the file. Lines 290–300 check to see if a particular group was chosen or if the whole class is to be printed. Lines 310–480 then print the first report. The student's R$ tally is separated using SEG$. Line 360 and line 410 are used to print information if only part of the ten weeks is used. If you have a different number of weeks in your report, you can change the 10 in lines 130, 410, 520, 560, 600, and 670, and the titles in lines 140 and 930–950.

Total Values

Variable names starting with T are total values. Lines 440–450 print total presentations divided by total possible weeks and the individual's percentage. Lines 500–630 print the totals for each week.

A bar graph report is printed in lines 640–700. Each asterisk represents a report, and the appropriate number of asterisks is printed for each week as a graph.

The final report in this program is to rank the students from high score to low score by percentage. Lines 720–780 contain the sort routine. The percentages were stored in the P array with the corresponding names in NN$. Lines 790–850 print the percents and names. Line 810 and the subroutine in lines 1000–1150 alphabetize the names of all students who have a zero score.

Console BASIC

You can, in fact, do file processing without Extended BASIC and all the peripherals. I used Extended BASIC mainly because of the ease in formatting the printing—lining up the columns. In regular console BASIC you can use subroutines to

line up columns of numbers and the TAB function to start the columns right. See my January 1984 column in COMPUTE! for some suggestions on formatting and screen scrolling.

To use a printer you need the RS-232 interface plus the printer. A number of different name brands of printers can be used with the TI-99/4A. The printer manuals should tell you what features the printer has and how to control different features, such as the number of characters per inch and form feeds. Using the printer and RS-232 manuals, you can determine the appropriate printer configuration necessary for the OPEN statement. Without a printer, you can print on the screen—just keep within the 28 print columns and print a screen at a time or use a scrolling delay method so you can read the information as it is printed.

To use a disk drive you also need the disk controller or disk controller card for the Peripheral Expansion box. The disk controller or card comes with a command module and a manual that describes disk procedures. To use a cassette, simply change the "DSK1.---" statements to "CS1" and change the VARIABLE to FIXED. The cassette system works fine—it just takes longer than the disk system.

Program 1: Birthday List

```
80 REM TI EXTENDED BASIC
90 REM DISK, PRINTER
100 REM BIRTHDAY LIST
110 CALL CLEAR
120 DISPLAY AT(12,5):"BIRTHDAY LIST
    "
130 OPTION BASE 1
140 DIM T$(140),BD(140),M$(12)
150 FOR I=1 TO 12 :: READ M$(I):: N
    EXT I
160 DATA JAN,FEB,MAR,APR,MAY,JUN,JU
    L,AUG,SEP,OCT,NOV,DEC
170 L=0 :: I=1 :: L$="---"
180 OPEN #1:"RS232.BA=600"
190 OPEN #3:"DSK1.SAMPLE",INTERNAL,
    INPUT ,VARIABLE 192
200 IMAGE "(3 SPACES)### ##
    (3 SPACES)####################
    ##  ########(3 SPACES)#########
    ################"
210 INPUT #3:G,N$,F$,A$,P$,BD(I),R$
    ,C$
220 IF C$="MOVED" THEN 210
230 IF N$="ZZZ" THEN 270
240 IF P$="" THEN P$="(4 SPACES)"
250 T$(I)=F$&" "&N$&"/"&P$&A$
260 I=I+1 :: GOTO 210
270 I=I-1 :: CLOSE #3
280 DISPLAY AT(23,1):"SORTING"
290 B=1
300 B=2*B :: IF B<=I THEN 300
310 B=INT(B/2):: IF B=0 THEN 360
320 FOR J=1 TO I-B :: C=J
330 D=C+B :: IF BD(C)<=BD(D)THEN 35
    0
340 AA=BD(C):: TT$=T$(C):: BD(C)=BD
    (D):: T$(C)=T$(D):: BD(D)=AA ::
     T$(D)=TT$ :: C=C-B :: IF C>0 T
    HEN 330
350 NEXT J :: GOTO 310
360 GOSUB 510
370 FOR J=1 TO I
380 IF BD(J)=0 THEN B$="---" :: D=0
     :: GOTO 420
390 DD$=STR$(BD(J)):: M=VAL(SEG$(BD
    $,1,LEN(BD$)-2)):: D=VAL(SEG$(B
    D$,LEN(BD$)-1,2))
400 B$=M$(M):: IF B$=L$ THEN 420
410 L=L+1 :: PRINT #1 :: L$=B$
420 P=POS(T$(J),"/",8)
430 N$=SEG$(T$(J),1,P-1)
440 P$="586-"&SEG$(T$(J),P+1,4)
450 A$=SEG$(T$(J),P+5,LEN(T$(J))-P+
    4)
460 PRINT #1,USING 200:B$,D,N$,P$,A
    $
470 L=L+1 :: IF L=48 THEN PRINT #1:
    CHR$(12):: L=0 :: GOSUB 510
480 NEXT J
490 PRINT #1:CHR$(12)
500 STOP
510 PRINT #1:TAB(34);"SAMPLE CLASS"
520 PRINT #1: :TAB(34);"BIRTHDAY LI
    ST"
530 PRINT #1: :TAB(33);"APRIL 15, 1
    984"
540 PRINT #1: : :TAB(5);"BIRTHDAY";
    TAB(15);"NAME";TAB(41);"PHONE";
    TAB(54);"ADDRESS"
550 PRINT #1:TAB(5);"--------";TAB(
    15);"----";TAB(41);"-----";TAB(
    54);"-------": : :
560 RETURN
570 END
```

Program 2: Report Writer

```
80 REM TI EXTENDED BASIC
90 REM DISK, PRINTER
100 REM REPORT WRITER
110 OPTION BASE 1
120 DIM D$(10),T(10),TT(10),NN$(140
    ),P(140)
130 FOR I=1 TO 10 :: READ D$(I):: N
    EXT I
140 DATA JAN 1,JAN 8,JAN 15,JAN 22,
    JAN 29,FEB 5,FEB 12,FEB 19,FEB
    26,MAR 4
150 DISPLAY AT(4,6)ERASE ALL:"REPOR
    T WRITER"
160 DISPLAY AT(7,3):"CHOOSE:" :: DI
    SPLAY AT(8,5):"1 GROUP 1" :: DI
    SPLAY AT(9,5):"2 GROUP 2"
170 DISPLAY AT(10,5):"3 GROUP 3" ::
     DISPLAY AT(12,5):"4 COMPLETE C
    LASS"
180 CALL KEY(0,KEY,ST)
190 IF KEY<49 OR KEY>52 THEN 180
200 G1=KEY-48 :: CALL HCHAR(7,3,32,
    192)
210 OPEN #1:"RS232.BA=600",VARIABLE
     132
220 REM SET FOR COMPRESSED PRINT
230 ESC$=CHR$(27):: PRINT #1:ESC$&"
    P"&"D"&ESC$&"\"
240 OPEN #3:"DSK1.SAMPLE",INTERNAL,
    INPUT ,VARIABLE 192
```

```
250 I=0 :: L$="^"
260 GOSUB 880 :: GOSUB 930
270 INPUT #3:G,N$,F$,A$,P$,BD,R$,C$
280 IF N$="ZZZ" THEN 490
290 IF G1=4 THEN 310
300 IF G1<>G THEN 270
310 IF SEG$(C$,1,5)="AUDIT" THEN 27
    0
320 C$=SEG$(N$,1,1):: IF L$<>C$ THE
    N L$=C$ :: PRINT #1 :: L=L+1
330 PRINT #1:TAB(10);N$;", ";F$;TAB
    (44);
340 TA=0 :: TP=0
350 IF R$="" THEN R$="00000000000"
360 FOR J=1 TO LEN(R$)
370 A$=SEG$(R$,J,1):: IF A$="1" THE
    N TA=TA+1

380 IF A$="1" OR A$="0" THEN TP=TP+
    1 :: T(J)=T(J)+VAL(A$):: TT(J)=
    TT(J)+1
390 PRINT #1:A$;"(4 SPACES)";
400 NEXT J
410 FOR JJ=J TO 10 :: PRINT #1:"
    (5 SPACES)";:: NEXT JJ
420 I=I+1 :: NN$(I)=F$&" "&N$
430 IF TP=0 THEN P(I)=0 :: GOTO 450
440 P(I)=INT(TA*100/TP)
450 PRINT #1,USING "(16 SPACES)##/##
    (5 SPACES)###":TA,TP,P(I)
460 L=L+1 :: IF L=48 THEN GOSUB 870
     :: GOSUB 930

470 IF A$="-" THEN I=I-1
480 GOTO 270
490 GOSUB 930
500 PRINT #1
510 PRINT #1:TAB(10);"REPORTS: ";TA
    B(42);
520 FOR J=1 TO 10
530 PRINT #1,USING "###  ":T(J);
540 TAT=TAT+T(J):: NEXT J
550 PRINT #1: :TAB(10);"ENROLLED: "
    ;TAB(42);
560 FOR J=1 TO 10
570 PRINT #1,USING "###  ":TT(J);
580 TE=TE+TT(J):: NEXT J
590 PRINT #1: : :TAB(10);"PERCENT R
    EPORTS:";TAB(42);
600 FOR J=1 TO 10
610 PRINT #1,USING "###  ":T(J)*100
    /TT(J);
620 NEXT J
630 PRINT #1:TAB(120);INT(TAT*100/T
    E)
640 GOSUB 870
650 PRINT #1: :TAB(10);"DATE";TAB(3
    0);"REPORTS"
660 PRINT #1:TAB(10);"----";TAB(30)
    ;"--------": :
670 FOR J=1 TO 10
680 A$=RPT$("*",T(J))
690 PRINT #1: :TAB(10);D$(J);TAB(30
    );T(J);"  ";A$
700 NEXT J
710 GOSUB 870
720 B=1
730 B=2*B :: IF B<=I THEN 730
740 B=INT(B/2):: IF B=0 THEN 790
750 FOR J=1 TO I-B :: C=J
760 D=C+B :: IF P(C)<=P(D) THEN 780
770 AA=P(C):: AA$=NN$(C):: P(C)=P(D
    ):: NN$(C)=NN$(D):: P(D)=AA ::
    NN$(D)=AA$ :: C=C-B :: IF C>0 T
    HEN 760
780 NEXT J :: GOTO 740
790 GOSUB 970
800 FOR J=I TO 1 STEP -1
810 IF P(J)=0 AND FL=0 THEN GOSUB 1
    000
820 PRINT #1:TAB(46);
830 PRINT #1,USING "###(8 SPACES)##
    ####################":P(J),N
    N$(J)
840 L=L+1 :: IF L=48 THEN GOSUB 870
     :: GOSUB 970
850 NEXT J
860 STOP
870 PRINT #1:CHR$(12)
880 PRINT #1:TAB(58);"SAMPLE CLASS"
890 IF G1=4 THEN 910
900 PRINT #1: :TAB(60);"GROUP";G1
910 PRINT #1: :TAB(53);"BOOK REPORT
    S PRESENTED"
920 PRINT #1: :TAB(57);"FIRST TERM
    1984" :: RETURN
930 PRINT #1: : :TAB(43);"JAN  JAN
     JAN  JAN  JAN  FEB  FEB  FEB
    FEB  MAR"
940 PRINT #1:TAB(10);"NAME";TAB(43)
    ;" 1(4 SPACES)8(3 SPACES)15
    (3 SPACES)22(3 SPACES)29
    (4 SPACES)5(3 SPACES)12
    (3 SPACES)19(3 SPACES)26
    (4 SPACES)4";TAB(110);"TOTAL";T
    AB(121);"%"
950 PRINT #1:TAB(10);"----";TAB(43)
    ;"---  ---  ---  ---  ---  ---
     ---  ---  ---  ---";TAB(110);"
    -----";TAB(120);"---": :
960 L=0 :: RETURN
970 PRINT #1: : :TAB(44);"PERCENT";
    TAB(57);"NAME"
980 PRINT #1:TAB(44);"-------";TAB(
    57);"----": : :
990 L=0 :: RETURN
1000 FOR K=1 TO J
1010 S=POS(NN$(K)," ",1)
1020 S1=POS(NN$(K)," ",S+1):: IF S1
     =0 THEN 1030 ELSE S=S1
1030 NN$(K)=SEG$(NN$(K),S+1,LEN(NN$
     (K))-S)&", "&SEG$(NN$(K),1,S-1
     )
1040 NEXT K
1050 B=1
1060 B=2*B :: IF B<J THEN 1060
1070 B=INT(B/2):: IF B=0 THEN 1120
1080 FOR K=1 TO J-B :: C=K
1090 D=C+B :: IF NN$(C)>=NN$(D)THEN
     1110
1100 A$=NN$(C):: NN$(C)=NN$(D):: NN
     $(D)=A$ :: C=C-B :: IF C>0 THE
     N 1090
1110 NEXT K :: GOTO 1070
1120 FOR K=1 TO J :: S=POS(NN$(K),"
     ",",",1)
1130 NN$(K)=SEG$(NN$(K),S+2,LEN(NN$
     (K))-S+1)&" "&SEG$(NN$(K),1,S-
     1)
1140 NEXT K
1150 FL=1 :: RETURN
1160 END
```

NEWS&PRODUCTS

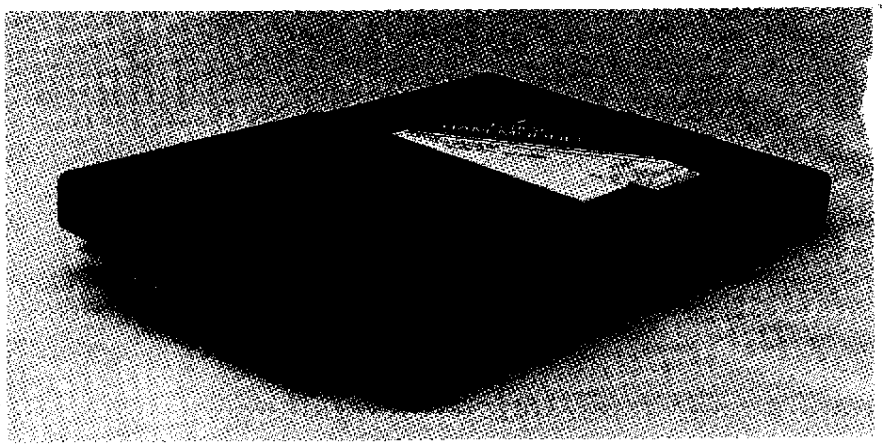
Memory Expander For VIC-20

Letco has announced the 64KV Memory Module, which adds more than 64K of memory to your VIC-20.

The 64KV houses 8K in each of the VIC's blocks 1, 2, and 3. Block 3 can also be paged, or swapped, under program control, with five other separate 8K sections of memory. Each block has a separate enable switch and a write-protect switch, and there is a switch to make block 3 respond as though it is block 5 (the normal game block).

The module is priced at $109.95

Letco
7310 Wells Road
Plain City, OH 43064
(614) 873-4410



The Letco 64KV Memory Module adds more than 64K RAM memory to your VIC-20.

Authoring System And Teaching Tool

CLAS, a teaching tool and authoring system for educators, has been released by Touch Technologies for the Apple II+ and IIe, the IBM PC and PCjr, and the Commodore 64.

The software package functions as a teaching tool for any subject. Authoring procedures allow instructors to create lessons in their own teaching style. Up to 30 problem sets can be offered with each lesson. Questions take the form of true/false, multiple choice, short answer, or matching.

If desired, the questions can be presented in a different order each time the lesson is used.

Sound is used to give feedback when a response is made to a question. A help mode is provided for the student, along with a review of problem areas and a summary of performance at the end of the lesson.

Memory requirement for Apple computers is 48K. The IBMs must use DOS 2.0/2.1. *CLAS* is available for $89.95.

Touch Technologies
609 S. Escondido Blvd.
Ste. 101
Escondido, CA 92025
(619) 743-0494

Interface For TI-99/4A

Mikel Laboratories, Inc., has announced an RS-232-C interface system for the TI-99/4A.

The $145.95 system is a free-standing unit which allows the TI-99/4A to use a printer and modem without a peripheral expansion unit.

The company also offers cassette interface systems ($49.95), TI cassette cables ($11.95), and printers and monitors. A line of personal computer accessories for the TI-99/4A will soon be available from Mikel Laboratories.

Mikel Laboratories
3341 W. El Segundo Blvd.
Hawthorne, CA 90250
(213) 679-2542

Life Insurance Program For Atari, Commodore

Advanced Financial Planning has released *Life Insurance Planning*, a software package for the Atari 400 and 800 computers and the Commodore 64.

The program will calculate the inflation rate applicable to a user's budget; the user's total estate needs reduced into terms of today's dollars (such as future living expenses for the family,