

BOGGLERS

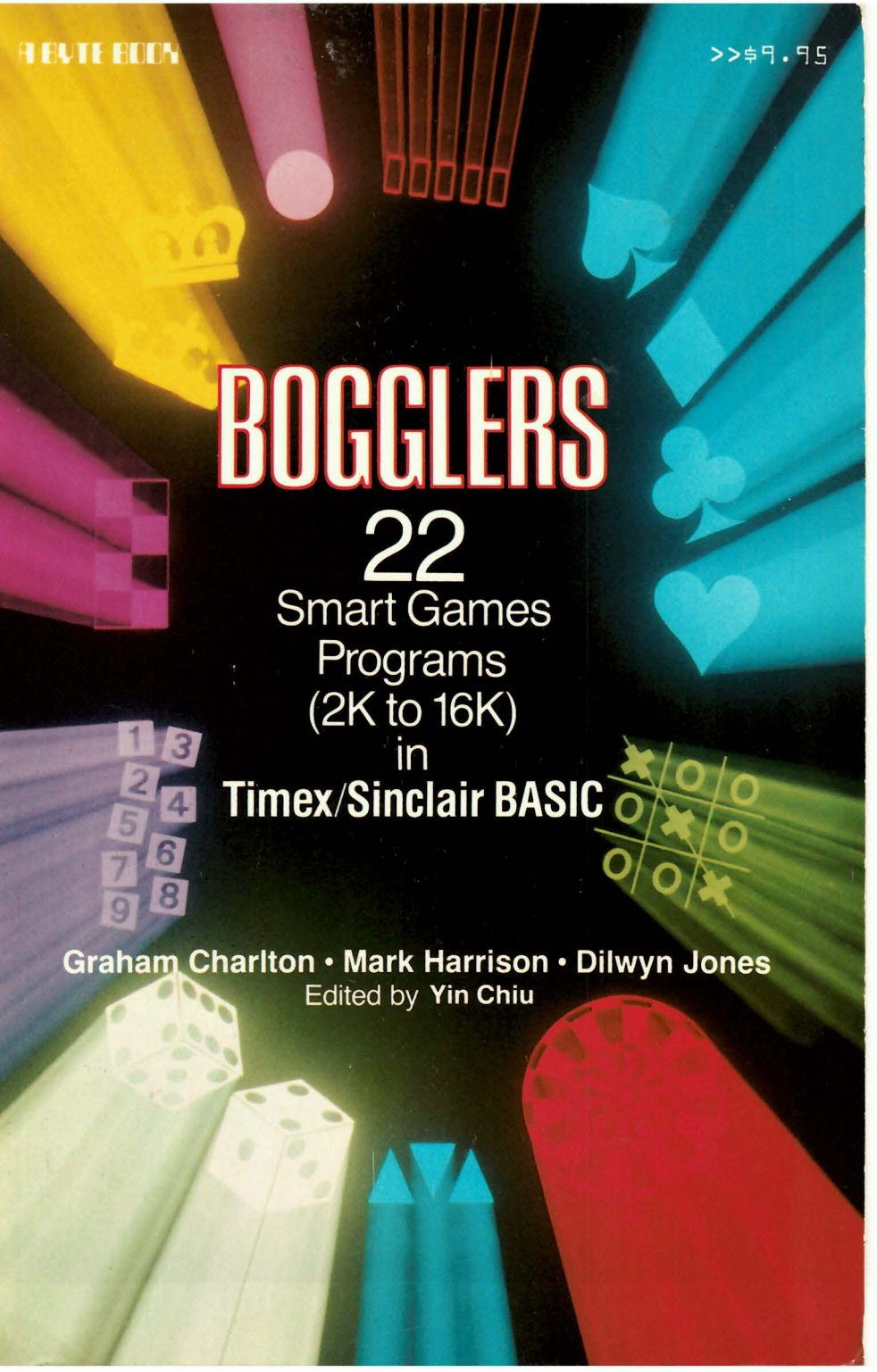
22

Smart Games
Programs
(2K to 16K)
in

Timex/Sinclair BASIC

Graham Charlton • Mark Harrison • Dilwyn Jones

Edited by Yin Chiu



BOGGLERS

BOGGGLERS

22

Smart Games

Programs

(2K to 16K)

in

Timex/Sinclair BASIC

Graham Charlton • Mark Harrison • Dilwyn Jones

Edited by Yin Chiu

McGraw-Hill Book Company

New York St. Louis San Francisco Auckland Bogotá
Guatemala Hamburg Johannesburg Lisbon London Madrid
Mexico Montreal New Delhi Panama Paris San Juan
São Paulo Singapore Sydney Tokyo Toronto

The authors of the programs provided with this book have carefully reviewed them to ensure their performance in accordance with the specifications described in the book. Neither the authors nor McGraw-Hill, however, makes any warranties whatever concerning the programs. They assume no responsibility or liability of any kind for errors in the programs or for the consequences of any such errors.

As used in this book, the terms "Timex/Sinclair 1000" and "T/S 1000" refer to the Timex/Sinclair 1000, a computer manufactured and sold by the Timex Computer Corporation. Timex/Sinclair 1000 is a registered trademark of the Timex Corporation.

BOGGLERS: 22 Smart Games Programs (2K to 16K) in Timex/Sinclair BASIC

Copyright © 1984 by Interface Publications. All rights reserved. Printed in the United States of America. Except as permitted under the United States Copyright Act of 1976, no part of this publication may be reproduced or distributed in any form or by any means, or stored in a data base or retrieval system, without the prior written permission of the publisher.

A BYTE Book.

1 2 3 4 5 6 7 8 9 0 S E M S E M 8 9 3 2 1 0 9 8 7 6 5 4 3

ISBN 0-07-023959-2

LIBRARY OF CONGRESS CATALOGING IN PUBLICATION DATA

Charlton, Graham.

Bogglers : 22 smart games programs (2K to 16K) in Timex/Sinclair BASIC.

(Byte books) (McGraw-Hill/VTX series)

1. Computer games. 2. Timex 1000 (Computer)—Programming. I. Harrison, Mark, date . II. Jones, Dilwyn. III. Chiu, Yin, date . IV. Title.
V. Series.

GV1469.2.C453 1984 794.8'2 83-18709

ISBN 0-07-023959-2

Contents

How to Use This Book	vii
-----------------------------	------------

Part 1 GAMES

1. Backgammon	3
2. Quatermass	16
3. Checkers	20
4. Sub Hunt	25
5. Gomoku	30
6. Biorhythms	34
7. Eliza	40
8. Four in a Row	57
9. Othello/Reversi	62
10. Fox and Geese	67
11. Pairs	70
12. Four Field Kono	75
13. Surf Rider	79
14. High Jump	82
15. Maze Master	87
16. Zombies	90
17. Poker Dice	93
18. Antimind	96

19. Nim	99
20. Hundred Up	102
21. Tic-Tac-Toe	105
22. Solitaire	114

Part 2 UTILITY AND APPLICATIONS PROGRAMS

23. Address Book	121
24. Resistor Code	125
25. Exam Results	133

APPENDIXES

1. Speeding Up Programs	141
2. Enlarged Characters	145
3. Useful System Variables	155
4. Decimal to Hexadecimal Converter	158
5. Hexadecimal to Decimal Converter	160
6. Special Graphics Symbols	162

How to Use This Book

Computing with Timex/Sinclair BASIC provides both entertainment and education for your entire family. The games and utility programs in this book will run on Timex/Sinclair machines with 16K of random access memory. All the programs have been fully tested, and once you enter them into your computer they will provide hours of enjoyment. Whether you challenge the computer to a game of Backgammon, discuss your problems with the computer psychiatrist Eliza, or compile an address book, you will find your Timex/Sinclair an intelligent ally, opponent, and assistant.

Even if you are already experienced in using the Timex/Sinclair computers, you should take a few minutes to read the following tips. Timex/Sinclair computers are remarkably powerful for machines which are also so inexpensive, but to use them to their best advantage you must be familiar with their unique dialect of BASIC, perhaps the most widely used computer language in the world, and you must be familiar with the peculiarities of the T/S keyboard, which is organized to make keyboarding as painless and efficient as possible.

A brief reading of the points covered here along with the instructions that appear in the *Timex User Manual* that came along with your computer should enable you to set up the computer and enter the first program within an hour or so.

Setting Up Your System

The *Timex User Manual* contains straightforward instructions for setting up your computer. We have found that a few ad-

ditional adjustments to the setup can make your system even easier to use.

1. *The cable connecting your computer to the TV.* For the most part the T/S computers produce a TV image with black characters on a white background, so the screen is quite bright. With the 4-foot cable supplied by the manufacturer to connect the computer and the TV, we found ourselves sitting almost on top of the screen, and after a few hours of work we were suffering from eye strain and a bit of lightheadedness. So we recommend that you replace the 4-foot cable with a 6- or 8-foot cable. Most electronics or television and radio stores will have such cables in stock, and one shouldn't cost you more than a few dollars. With a longer cable, we were able to move the TV to the far end of a table. In such a setup, we still found the image on the screen to be perfectly readable, and it was certainly more restful for our eyes.
2. *The channel switch on the bottom of the computer.* At times when we've been programming or playing a game on the Timex/Sinclair, we've slid the computer a few inches across the table to let another person take the controls. Sometimes when we've done this, we've completely lost the image on the screen. A little investigative work turned up the answer: the channel-selection switch on the bottom of the computer is supposed to be recessed into the machine, but on some of the computers we've dealt with it is not recessed enough. As a result, sliding the computer across a surface can throw the switch from one channel to the other, and there will be a temporary loss of the picture. This is not something to worry about. By simply flicking the switch back to the proper channel position, you should find that the image comes back onto the screen with no loss of whatever has been stored in memory. If this problem plagues your machine, you might want to consider "locking" the channel selection switch in place with a piece of tape.

We have also found that the manufacturer's advice about whether to use channel 2 or channel 3 does not always seem to be the best advice. Of the three machines we tested, two worked best on channel 3 and the third worked best on channel 2. Since all three machines were tested in New York City, where the local CBS-TV affiliate broadcasts on channel 2, we assumed that channel 3 would work best for us, but this was not always the case. We can't explain this fact, but we want you to be aware of it.

3. *Setting up your tape recorder.* In saving programs and loading programs, we've found it best to have the "volume" control of the tape recorder at or near its loudest setting, as mentioned in the *Timex User Manual*. Similarly, we've had the best results in loading and saving when the "tone" control is at its highest setting—toward "treble" rather than "bass." Furthermore, if your tape recorder has both "monaural" and "stereo" modes, use the "monaural" setting. If a stereo setting is all that is available, you should turn the "balance" control all the way to the left or all the way to the right and leave it there for both loading from and saving to the tape recorder.

The *Timex User Manual* also contains a very clever suggestion about how to store programs in such a way that they are easy to find: see page 14 of the manual for instructions on how to insert a voice announcement preceding each of the programs you store on tape. Be sure, however, to disconnect the microphone before you save the actual program from the computer to the tape.

Three further points: First, we've found that it pays to use a fairly high quality recording tape. Second, it is a good idea to make a backup copy of each program, preferably on a completely different cassette. And third, you should record programs only on one side of any tape, since if both sides are used, the signals on one side may interfere with those on the other.

4. Perhaps the greatest shortcoming of the Timex/Sinclair is its relatively small keyboard. For maximum comfort, convenience, and accuracy of keyboarding, keep the keyboard well lighted and (as we have already mentioned) a good distance away from that glaring screen. With a few minutes of practice, you'll find that the keyboard is indeed manageable.

Keyboarding Hints

If you haven't already had some experience with the T/S keyboard, you should read this section carefully and keep the keyboard close by for reference.

1. When you are typing in the number 1, be sure to hit the 1 key and not the L key. The 1 key is located on the top row of the keyboard at the extreme left.
2. Don't confuse the number 0 with the letter O. In program listings in this book and on the screen of your TV, you'll notice that zero is printed with a slash through it, like this: 0.
3. The letter I and the number 1 may also look alike to you as you read the program listings, but you must be sure to distinguish between them. The letter I is shown in the programs this way: I. The number 1 is listed in the programs this way: 1.
4. In some of the programs we make use of an underscore (_) to indicate that you should leave a space (by hitting the SPACE key). Do not attempt to key in the underscore, because you will not find it on the T/S keyboard. We make use of it only within quotation marks in the program listing because it is only within quotation marks that inserting the proper number of spaces may be crucial.
5. All words printed in **BOLDFACE** in the program listings are called keywords. Each of these words is accessed by a single keystroke. If you try to type in one of these words letter-by-letter, the computer will either not accept them or will not run the program

correctly. So whenever you see a boldface word in a listing, be sure to access it with a single keystroke.

6. All of the following mathematical symbols are also accessed with a single keystroke:

< > which means “not equal to” and is found on the T key.

> = which means “greater than or equal to” and is found on the Y key.

< = which means “less than or equal to” and is found on the R key.

7. Keep in mind that any given key on the T/S keyboard may have as many as five meanings, depending on the *mode* you are in. The mode is indicated by a cursor, as follows:

K which indicates the keyword mode. When you are in this mode, you will access the word printed in white above the key.

L which indicates the letter mode. When you are in this mode, you will access the “primary” (dark black) character on the key.

F which indicates the function mode. When you are in this mode, you will access the word printed in white below the key.

G which indicates the graphics mode. If you hit a key in this mode, the “primary” (dark black) character will appear in inverse video. (This means that the character will appear on the screen as a white number or letter printed in a small black box.) If you hold down the SHIFT key while in the graphics mode, some of the keys will give you the small graphics symbols shown at the lower right corner of the key.

Saving Programs on a Tape Recorder

After you've keyed in a program, checked it for accuracy, and decided you want to keep it handy, you can save it on magnetic tape in much the same way that you ordinarily save a piece of music by recording it on your tape recorder. All you have

to do is to type the word **SAVE** and decide on a name for your particular program. If you choose the name **SUB HUNT**, for example, you would type into your computer:

SAVE "SUB HUNT"

At this point you should start the recorder in its "recording" mode. Then hit the **ENTER** key, and the program (or more accurately, a copy of your program, since the program will still reside in the memory of the computer) will be recorded on the tape. You will know when the recording is complete by the termination code \emptyset/\emptyset which will appear at the bottom left-hand corner of the screen. At that point you may turn off the recorder.

Loading a Program

In order to load a program from magnetic tape into the computer, type in **NEW** followed by the **ENTER** key. Now you type either:

LOAD "SUB HUNT"

which searches for and loads the program called **SUB HUNT**, or:

LOAD " "

which searches for and loads the first complete program found on the tape.

Important Reminders

1. Make a note of the *precise* name of the program you are storing, including any spaces between words that may exist in the name. If you save a program called **SUB HUNT** on tape and then later try to load a program called **SUBHUNT** from the same tape, the computer will *not* be able to find the program.
2. We have found that for either loading or saving a program, a tape recorder in the "mono" (rather than "stereo") mode works best. If your tape recorder is

stereo only, you should turn the “balance” control all the way left or all the way right. In addition, you should leave the “volume” control quite high, perhaps three-quarters of the way to full volume; the “tone” control should likewise be at its highest (treble) setting.

1

Part

GAMES

Backgammon

This program allows the computer to challenge you to single games of Backgammon. The program insists on fully legal moves and will reject any attempts at cheating.

<--TS1000 MOVES



<--YOUR MOVES

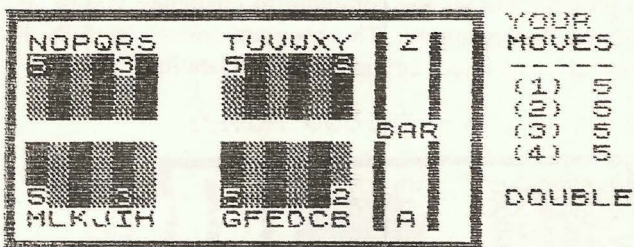
The board display is limited a little by the computer's graphics capability. For example, the normally triangular "points" are represented by rectangles. The individual pieces are not shown. Instead, the number of pieces on each point is shown as the numbers 0 to 9, or if there are more than nine pieces, by the letters A to F, which indicate the numbers 10 through 15 as follows:

A = 10; B = 11; C = 12; D = 13; E = 14; and F = 15.

These are the same as hexadecimal notation, and you get used to it surprisingly quickly. Anyway, it's only rarely you will have more than nine pieces on a point.

The computer's pieces are shown as inverse characters (white characters on a black background) and move from point Y toward point B. The computer bears off onto point A. Any computer pieces which are "hit" are returned to the bar at point Z. You move your "ordinary video" pieces from point B toward point Y, bearing off to point Z. Any of your pieces hit are returned to the bar at point A.

The program uses its own dice, and the moves are shown to the right of the board. Normally, two numbers representing the dice thrown are shown. If you throw a "double," four numbers are shown, along with a message to remind you that you've thrown a double:



If you find that at any time you can't move (for example, if you have a piece on the bar and it cannot be used) then simply press ENTER when the computer asks for your moves. The computer will check that you cannot move (in case you've made a mistake, or cheated), then will either make its move or tell you to make a legal move if there is one.



YOU CAN MAKE A LEGAL MOVE

If at any time you wish to stop the program, press **BREAK** if it is the computer's move, or enter **STOP** (shifted-A) if it is your move. The program will automatically detect when a game has been won or lost. To enter your move, you need to enter the dice thrown chosen, and the point from which you are moving—that is, a letter and a number. These may be entered in any order. They will be sorted by the computer. For example, if you had thrown a three and wished to move one of your pieces from point B, you could enter either "B3" or "3B". If the move is permitted, you will see the pieces move on the board. Otherwise, you'll have to re-enter your move.

Program 1: Backgammon

```

10  REM BACKGAMMON
20  REM RUNS IN 6K
30  REM INITIALIZE
50  GOSUB 9000
60  REM BY DILWYN JONES 1982
999 REM PLAYER MOVES
1000 PRINT AT 0,23;"YOUR_ _ _"†
1005 LET B$="1"
1009 REM ROLL DICE
1010 GOSUB 2000
1030 DIM M$(2)
1040 INPUT M$
1042 RAND
1045 IF M$(1)="_STOP_" THEN STOP
1050 IF M$="_ _" THEN GOTO 1200

```

† Underscores within quotation marks show the number of spaces you must leave.

```

1055  IF M$(1) >="1" AND
      M$(1) <="6" AND M$(2) >="A" AND M$(2)
      <="Y" THEN LET M$=M$(2)+M$(1)
1060  IF M$(1) <"A" OR M$(1) >"Y" OR M$(2) <"1"
      OR M$(2) >"6" THEN GOTO 1040
1070  FOR G=1 TO D
1080  IF D$(G)=M$(2) THEN GOTO 1110
1090  NEXT G
1100  GOTO 1040
1110  LET FROM=CODE M$-37
1120  LET TO=FROM+VAL M$(2)
1125  IF FROM <> 1 AND A$(2,1) <> A$(1,1) THEN
      GOTO 1040
1130  IF A$(2,FROM) <"1" OR A$(2,FROM) >"F" THEN
      GOTO 1040
1135  IF TO < 26 THEN IF A$(2,TO) = "1" THEN GOTO
      3000
1140  IF TO < 26 THEN IF A$(2,TO) >="2" AND
      A$(2,TO) <="F" THEN GOTO 1040
1150  IF TO < 26 THEN GOTO 4900
1155  FOR H=1 TO 19
1160  IF A$(2,H) >="1" AND A$(2,H) <="F" THEN
      GOTO 1040
1165  NEXT H
1180  GOTO 4000
1198  REM PLAYER NULL INPUT
1199  REM CAN PLAYER MOVE?
1200  LET FLAG=0

```



```

1530 FOR F=26 TO 2 STEP -1
1540 LET FROM=F
1545 IF FROM <= 7 AND FLAG=0 AND NOT
    BEAROFF THEN GOTO 1700
1550 IF A$(2,FROM) < "1" OR A$(2,FROM) > "F"
    THEN GOTO 1650
1560 FOR G=1 TO D
1570 IF D$(G)="_" THEN GOTO 1620
1580 LET TO=FROM-VAL D$(G)
1590 IF TO < 2 AND FLAG=0 THEN GOTO 4000
1600 IF TO > 1 THEN IF A$(2,TO)="1" THEN GOTO
    3000
1610 IF A$(2,TO) < "1" OR A$(2,TO) > "F" THEN GOTO
    4900
1620 NEXT G
1630 IF A$(2,26) < > A$(1,26) THEN GOTO 1000
1640 IF FLAG=0 THEN LET FLAG=1
1650 NEXT F
1660 GOTO 1000
1699 REM TS1000 BEARING OFF
1700 FOR H=7 TO 2 STEP -1
1710 IF A$(2,H)=A$(1,H) THEN GOTO 1780
1720 LET FROM=H
1730 FOR G=1 TO D
1740 IF D$(G)="_" THEN GOTO 1770
1750 LET TO=FROM-VAL D$(G)
1760 IF TO < 2 THEN GOTO 4000
1770 NEXT G

```

```

1780 NEXT H
1785 LET F=7
1787 LET BEAROFF=1
1790 GOTO 1540
1999 REM DICE SUBROUTINE
2000 DIM D$(4)
2005 PRINT AT 8,23;" _ _ _ _ _"
2010 LET D$(1)=STR$(INT (RND*6)+1)
2020 LET D$(2)=STR$(INT (RND*6)+1)
2030 IF D$(1)=D$(2) THEN LET D$(3 TO 4)=D$(1 TO 2)
2040 LET D=2+(2 AND D$(1)=D$(2))
2050 PRINT AT 3,27;D$(1);TAB 27;D$(2);TAB
      27;D$(3);TAB 27;D$(4)
2070 IF D=2 THEN RETURN
2079 REM DOUBLE THROW
2080 FOR F=1 TO 20
2090 PRINT AT 8,22;"DOUBLE";AT 8,22; " _
      DOUBLE_"
2100 NEXT F
2110 RETURN
3000 REM HIT A BLOT
3010 LET A$(2,TO)=B$
3020 LET A$(2,FROM)=(A$(1,FROM) AND
      A$(2,FROM)=B$)+(CHR$(CODE A$(2,FROM)-1)
      AND A$(2,FROM)>B$)
3030 IF B$="1" THEN LET A$(2,26)="1" AND
      A$(2,26)=A$(1,26)+(CHR$(CODE A$(2,26)+1)
      AND A$(2,26)>="1")

```

```

3040  IF B$="1" THEN LET A$(2,1)="1" AND
      A$(2,1)=A$(1,1))+(CHR$(CODE A$(2,1)+1) AND
      A$(2,1)>="1")
3050  GOTO 5000
3999  REM BEARING OFF
4000  IF B$="1" THEN LET P$=CHR$(CODE P$+1)
4010  IF B$="1" THEN LET Z$=CHR$(CODE Z$+1)
4020  LET A$(2,FROM)=(A$(1,FROM) AND
      A$(2,FROM)=B$)+(CHR$(CODE A$(2,FROM)-1)
      AND A$(2,FROM)>B$)
4040  IF B$="1" AND P$>"0" THEN PRINT AT
      3,18;P$
4045  IF B$="1" AND Z$>"0" THEN PRINT AT
      7,18;Z$
4050  GOTO 5000
4899  REM SET FROM/TO POINTS
4900  IF TO>1 AND TO<26 THEN LET A$(2,TO)=(B$
      AND A$(2,TO)=A$(1,TO))+(CHR$(CODE
      A$(2,TO)+1) AND A$(2,TO)>=B$)
4910  LET A$(2,FROM)=(A$(1,FROM) AND
      A$(2,FROM)=B$)+(CHR$(CODE A$(2,FROM)-1)
      AND A$(2,FROM)>B$)
5000  REM PRINT MOVES, DICE, WON?
5010  LET D$(G)="_"
5020  PRINT AT 2+G,27;D$(G)
5030  PRINT AT CODE Y$(FROM),CODE
      X$(FROM);A$(2,FROM)
5040  IF TO>1 AND TO<26 THEN PRINT AT CODE
      Y$(TO),CODE X$(TO);A$(2,TO)

```

```

5050 PRINT AT CODE Y$(1), CODE X$(1);A$(2,1);
      AT CODE Y$(26), CODE X$(26); A$(2,26)
5060 IF P$="F" OR Z$="F" THEN GOTO (6000
      AND P$="F")+ (6500 AND Z$="F")
5070 IF D$="_ _ _ _" THEN GOTO (1000 AND B$=
      "1")+ (1500 AND B$="1")
5080 GOTO (1030 AND B$="1")+ (1540 AND B$="1")
6000 REM PLAYER HAS WON
6010 FOR F=1 TO 50
6020 PRINT AT 16,10;"YOU WIN"; AT 16,10;" YOU_
      WIN "
6030 NEXT F
6035 IF RND<.3 THEN PRINT AT 18,8; "YOU_
      WERE_LUCKY"
6040 STOP
6500 REM TS1000 HAS WON
6510 FOR F=1 TO 50
6520 PRINT AT 16,11; "I WIN";AT 16,11;" I_WIN "
6530 NEXT F
6535 IF RND<.3 THEN PRINT AT 18,8; "TS1000_
      RULES, OK?"
6540 STOP
8999 REM INITIALIZE
9000 CLS
9001 PRINT AT $4,10;"■■■■■■■■■■■
      ■■■";TAB 10; "■■BACKGAMMON■■";TAB
      10; "■■■■■■■■■■■"
9002 PRINT AT 10,15; "BY"; AT 12, 10; "DILWYN_
      JONES"

```

12 / Chapter 1

```
9005 FOR F=1 TO 70
9006 NEXT F
9008 CLS
9010 PRINT " ████████████████████████████████████████
██████████████████████"
9020 PRINT " █████NOPQRS _ _ _ TUVWXYZ _██z██_
██"
9030 PRINT " █████5███3███ _ _ _ 5███
2_██_██_"
9040 PRINT " ████████████████████████████████████████
██_██_██_"
9050 PRINT " ████████████████████████████████████████
██████_██_██_"
9060 PRINT " █████ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
_ BAR_██"
9070 PRINT " ████████████████████████████████████████
██████_██_██_"
9080 PRINT " ████████████████████████████████████████
██████_██_██_"
9090 PRINT " █████5████3███ _ _ _ 5██████2
_██_██_"
9100 PRINT " █████MLKJIH_ _ _ GFEDCB_██A██_██"
9110 PRINT " ████████████████████████████████████████
██████████████████████"
9115 DIM A$(2,26)
9120 LET A$(1)=" ████████████████████████████████████████
██████████████████████████████████████████ _"
9130 LET A$(2)=" _2██████53██████55██████
35██████2_"
```



```

9140 LET Y$="███████████████████████████████████
          ████"
9150 LET X$=">?:$£"+CHR$11+"███▣▣▣▣▣▣▣▣▣▣
          ████▣▣▣▣▣▣▣▣▣▣███"+CHR$11+"£$:?>"
9160 PRINT AT 1,23;"MOVES";TAB 23;"-----";
      TAB 23;"(1)";TAB 23;"(2)";TAB 23;"(3)";TAB
      23;"(4)"
9170 LET P$="∅"
9180 LET Z$="∅"
9200 RETURN
9900 SAVE "BACKGAMMON"
9910 RUN

```

Programming Notes

The program has been written to speed along in the **SLOW** mode, without resorting to machine code. Most moves are made in a second or two, perhaps rising to five seconds for difficult moves. To enable the program to run this quickly, it was necessary to add all sorts of markers and other facilities. This contributed significantly to the length of the program. A great deal of memory is also taken up by the display routines, error trapping statements, and the copious **REM** statements. If these were simplified, users of 4K systems could probably use this program, since the "core" takes about 3K.

The program has a liberal sprinkling of **REM** statements, so only a brief explanation is needed at this point.

A\$ is the main array. A\$(1) contains the state of the board at points with no pieces. A\$(2) holds the state of the board points A to Z. Note how well Backgammon lends itself to all the letters of the alphabet! Y\$ contains characters whose **CODE** corresponds to the Y coordinate of the Point under consideration and is used in the printing routines. X\$ similarly

contains the X coordinates. P\$ and Z\$ hold the number of pieces borne off by the player and by the computer, respectively.

M\$ is the string containing the player's move. D\$ is the string that holds the dice throws. FROM is the variable referring to the point from which the move is being made, and TO (which is the letters T and O, incidently, *not* the function TO on shifted-4) obviously refers to the square to which the move is being made. These are the most important variables.

The block of code starting at line 1000 accepts the player's move and decides if it is a legal move. The block at 1200 scans the board if the player merely presses ENTER when entering a move to see if, in fact, a move can be made. Line 1500 is the start of the block handling the computer's move. The block at 1700 looks after the computer's bearing off if the rules of the game actually permit the computer to do so (no computer pieces on the bar, and all computer pieces either borne off or on the computer inner board on points G to B).

The block at line 2000 is a subroutine generating the dice throws. The dice throws are stored as characters from "1" to "6". Normally there are only two, but if the throw was a double (both dice the same), D\$ is given four dice values. D is the variable that says how many numbers are in D\$. Lines 2080, 2090 and 2100 flash the word DOUBLE in inverse and normal video alternately on the screen to bring it to the player's attention. The routine at line 3000 takes appropriate action if a blot is hit. Note how B\$ performs both the function of telling the computer whose move it is and also what character will fill a point if a move is made to an empty point.

The next block handles bearing off. The appropriate checks on whether a player is allowed to bear off will have been made before the program gets to here. The block starting at line 4900 sets the FROM and TO points to appropriate values. Line 5000 is the start of the routine that sets the display and updates both the board and the dice throws remaining.

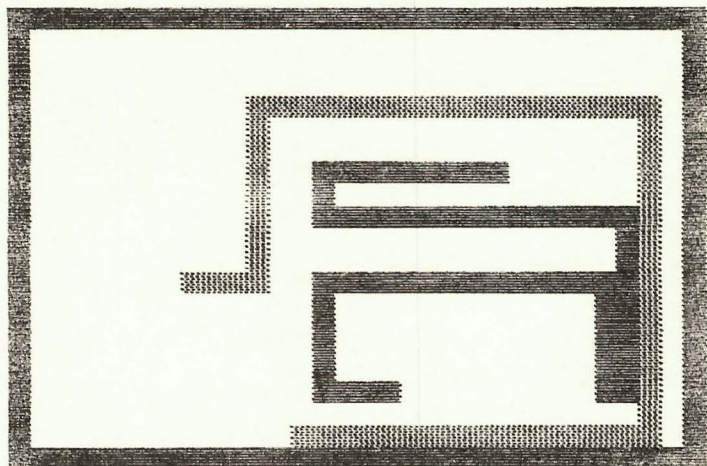
Lines 6000 to 6540 are the "victory" routines. This is where the program ends if it decides that one of the players

has won. The subroutine at line 9000 initializes the program, display, and the like. Lines 9900 and 9910 form a **LOAD** and **RUN** routine. When you've typed in the program, **SAVE** it on tape with the command **RUN 9900**. The program will then start up automatically when loaded from tape.

2

Quatermass

The object of this game is to surround your enemy, who is in turn trying to surround you. Press "W" to move up, "X" to move down, "A" to move right and, "D" to move left. You are the checkerboard trail, and the computer is the solid black trail.



Program 2: Quatermass

25 SLOW

30 PRINT AT 0,4;"QUATERMASS.";AT 1,4;"■■■■"
■■■■■■■■";AT 3,4;"THE_OBJECT_OF_

```

THE_GAME_IS_TO_SURROUND_YOUR_
ENEMY_WHO_IN_TURN_IS_ALSO_TRYING_
TO_SURROUND_YOU."
35 PRINT AT 7,0;"PRESS:","""W""_TO_MOVE_
UP","""X""_TO_MOVE_DOWN","""A""_TO_
MOVE_RIGHT","""D""_TO_MOVE_LEFT"
40 PRINT AT 13,0;"PLAYERS_COLOR_█ █ █ █";
AT 15,0;"ENEMYS_COLOR_█ █ █ █"
45 PRINT AT 21,0;"PRESS_ENTER_TO_BEGIN."
50 INPUT QS
55 CLS
60 RAND
65 LET A$="0010-1033-330010-1033"
70 FOR I=0 TO 31
75 PRINT AT 0,I;"█";AT 20,I;"█";AT I AND I
<20,31;"█ █"
80 NEXT I
85 LET D=255*PEEK 16397+PEEK 16396
90 LET X=D+405
95 LET Z=D+254
100 LET ZD=-1
105 LET XD=1
110 POKE X,8
115 POKE Z,128
120 IF INKEY$="" THEN GOTO 120
125 LET D=(33 AND INKEY$="X")-(33 AND
INKEY$="W")+ (1 AND INKEY$="D")-(1 AND
INKEY$="A")
130 IF D<>0 THEN LET XD=D

```

18 / Chapter 2

```
135  LET X=X+XD
140  IF PEEK X < > 0 THEN GOTO 230
145  POKE X,8
150  IF PEEK (Z+ZD)+PEEK (Z+2*ZD)+PEEK
    (Z+3*ZD)=0 AND RND < 0.9 THEN GOTO 215
155  LET D=3*INT (RND*4)+1
160  FOR I=3 TO 1 STEP -1
165  FOR J=D TO D+9 STEP 3
170  LET ZD=VAL A$(J TO J+2)
175  FOR K=1 TO I
180  IF PEEK (Z+K*ZD) < > 0 THEN GOTO 195
185  NEXT K
190  GOTO 215
195  NEXT J
200  NEXT I
205  PRINT AT 21,0;"YOU_WIN"
210  GOTO 235
215  LET Z=Z+ZD
220  POKE Z,128
225  GOTO 125
230  PRINT AT 21,0;"TS1000_WIN"
235  INPUT Q$
240  IF Q$="" THEN GOTO 55
245  STOP
```

Sample Run

QUATERMASS



THE OBJECT OF THE GAME IS TO
SURROUND YOUR ENEMY WHO IN TURN
IS ALSO TRYING TO SURROUND YOU.

PRESS:

"W" TO MOVE UP

"X" TO MOVE DOWN

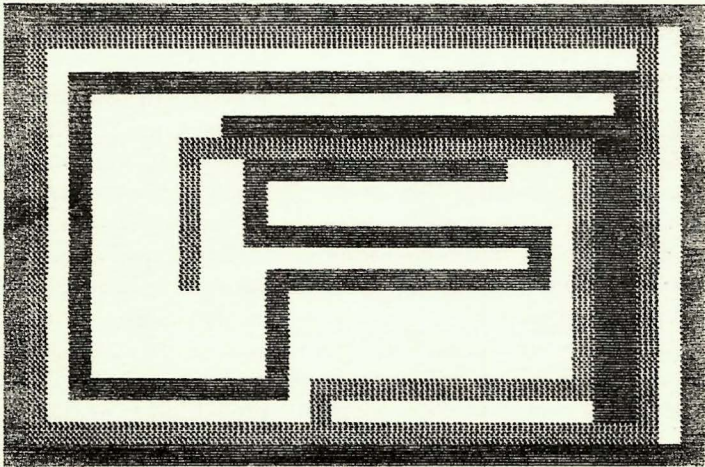
"A" TO MOVE RIGHT

"D" TO MOVE LEFT

PLAYERS COLOR 

ENEMYS COLOR 

PRESS ENTER TO BEGIN.



TS1000 WIN

Checkers

This program puts up a surprisingly tough game of CHECKERS. You'll need to concentrate to defeat it.

The string in line 4000 prints out the board, with line 4010 reading the odd collection of characters you see, and from their codes gets information on the position of a piece in the string A\$. These are loaded into the array in lines 4030 to 4050, so the computer knows where its pieces are all the time. 4060 contains information which the computer uses with information from line 1040 and similar ones to look for squares that are related diagonally.

Lines 1000 to 2990 contain the computer's decision-making algorithm. 1010 to 1170 checks each piece to see if it can capture. If it finds a possible capture, it makes it. This part of the algorithm is fairly slow at the start of the game when the computer has a lot of pieces on the board, but towards the end of the game its move decisions are made almost instantly. Multiple jumps are also made within this routine (a relatively simple thing to achieve, as you'll see), although a slowing down loop may be needed in order to see it! Line 1120 checks to see if a piece should be promoted to king, and if it finds one, sends control to 2990 where the promotion is made.

Lines 2000 and 2130 are for the non-capture move. The computer first searches to see if it can move a piece where it will not be in danger of capture, and if it does not find one, looks for a legal move, even if it places one of its pieces in

danger. If no legal move is found, then YOU WIN message (line 2140) is printed.

Lines 3000 to 3230 accept the player's move. The move is entered in two strings, G\$ and H\$, each containing a letter from A to H, and a number from 1 to 8. The first string is the square you are playing from, and the second is the square you are playing to. These are printed out with "?" after them, asking if this is correct. If not, enter "N", or enter anything if the move is correct.

If you capture a piece, the board will be reprinted, and you'll be asked if you can jump again. If you can, enter "Y".

Program 3: Checkers

```

10  GOTO 4000
100  PRINT AT 6,11;
110  FOR A=1 TO 91 STEP 10
120  PRINT TAB 11;A$(A TO A+9)
130  NEXT A
140  RETURN
1000  GOSUB 100
1010  LET Q=0
1020  FOR B=1 TO 12
1022  LET C=B(B)
1025  IF A$(C) < > "X" AND A$(C) < > "$" THEN GOTO
      1170
1030  FOR X=1 TO 4
1040  LET N=CODE B$(X)-50
1050  IF NOT ((X < 3 AND A$(C)="X") OR A$(C)=
      "$") THEN GOTO 1150
1060  IF NOT (A$(C+N)="O" OR A$(C+N)="E")
      THEN GOTO 1150
1070  IF NOT (A$(C+2*N)="_" THEN GOTO 1150

```

```

1080 LET A$(C+2*N)=A$(C)
1090 LET A$(C)="_"
1100 LET A$(C+N)="_"
1105 LET B(B)=C+2*N
1110 LET C=C+2*N
1120 IF C>80 THEN GOTO 2990 ]only if main -FD
1125 GOSUB 100
1130 LET Q=1
1140 GOTO 1030
1150 NEXT X
1160 IF Q=1 THEN GOTO 3010
1170 NEXT B
2000 FOR A=1 TO 2
2010 FOR B=1 TO 12
2015 LET C=B(B)
2017 IF A$(C)<>"X" AND A$(C)<>
"$" THEN GOTO 2120
2020 FOR X=1 TO 4
2030 LET N=CODE B$(X)-50
2040 IF NOT ((X<3 AND A$(C)="X") OR A$(C)="$")
THEN GOTO 2120
2050 IF NOT (A$(C+N)="_" ) THEN GOTO 2110
2060 IF A=1 AND ((A$(C+2*N)="O" OR
A$(C+2*N)="E" ) OR ((A$(C+N+20-ABS
N)="O" OR A$(C+N+20-ABS N)="E" ) AND
A$(C+N-20+ABS N)="_" )) THEN GOTO
2110
2070 LET A$(C+N)=A$(C)
2080 LET A$(C)="_"

```

```

2085 LET B(B)=C+N
2090 IF C+N>80 THEN LET A$(C+N)="$"
2100 GOTO 3000
2110 NEXT X
2120 NEXT B
2130 NEXT A
2140 PRINT AT 17,12;"YOU WIN";Z
2990 LET A$(C)="$"
3000 GOSUB 100
3010 PRINT AT 17,8;"FROM_?_"
3020 INPUT G$
3030 PRINT AT 17,13;G$;"_TO_?_"
3040 INPUT H$
3050 PRINT AT 17,19;H$;"_?"
3060 INPUT Z$
3080 PRINT AT 17,8;"_ _ _ _ _"
    _ _"
3090 IF Z$="N" THEN GOTO 3000
3100 LET G=10*(CODE G$(1)-37)+CODE G$(2)-27
3110 LET H=10*(CODE H$(1)-37)+CODE H$(2)-27
3120 LET A$(H)=A$(G)
3130 IF H<20 THEN LET A$(H)="$"
3140 LET A$(G)="_"
3150 IF ABS(H-G)<>22 AND ABS(H-G)<>18
    THEN GOTO 1000
3160 LET A$((H+G)/2)="_"
3170 GOSUB 100
3180 PRINT AT 17,6;"CAN_YOU_JUMP_AGAIN?"
3190 LET Z$=INKEY$

```

] wrong
 F. D.

24 / Chapter 3

```

3200  IF Z$="" THEN GOTO 3190
3210  PRINT AT 17,6;" _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
    _ _ _ _ _ _ _ _ "
3220  IF Z$="Y" THEN GOTO 3010
3230  GOTO 1010
4000  LET A$="12345678ABCDEFGHIJKL
    MNOPQRSTUVWXYZ _ _ _ _ _
    _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
    _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
    _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
    _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
    _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
    12345678"
4010  LET B$="$<?), -0/B759"
4020  DIM B(12)
4030  FOR B=1 TO 12
4040  LET B(B)=CODE B$(B)
4050  NEXT B
4060  LET B$="XVDB"
4070  PRINT AT 2,2;"DO_YOU_WANT_TO_GO_
    FIRST?_Y/N"
4080  LET Z$=INKEY$
4090  IF INKEY$="" THEN GOTO 4080
4100  PRINT AT 2,2;"CHECKERS_BY_GRAHAM_
    CHARLTON "
4110  IF Z$="Y" THEN GOTO 3000
4120  GOTO 1000

```

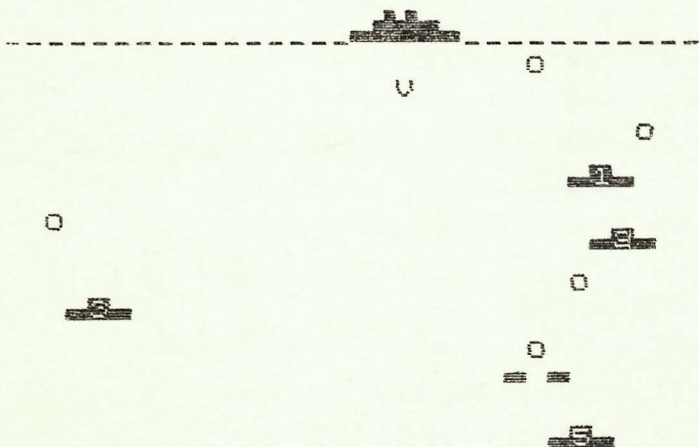
4

Sub Hunt

You are the captain of a destroyer, and it is your task to seek and destroy five enemy submarines.

You can drop one depth charge at a time. Each submarine is capable of releasing a bomb which floats upward, sinking any ship it hits.

Your controls are the "M" and "N" keys. "B" will drop a depth-charge. Your time is displayed on the screen if you survive. A "best time" feature also operates this game.



Program 4: Sub Hunt

```

5  GOSUB 500
10  LET L=1E9
15  CLS
20  DIM A(5)
25  DIM B(5,2)
30  FOR I=1 TO 5
35  LET A(I)=INT (RND*57-28)
40  LET B(I,1)=2+3*I
45  LET B(I,2)=ABS A(I)+2
50  NEXT I
55  LET Z=15
60  LET N=0
65  LET B=0
70  FOR I=0 TO 31
75  PRINT AT 1,I;"-"
80  NEXT I
85  POKE 16437,255
90  POKE 16436,255
95  FOR I=1 TO 5
100 PRINT AT 0,Z;" _ _ ■■■ _"; AT 1,Z;"-■
    ■■■■ _"
101 IF A(I)>98 THEN GOTO 115
102 PRINT AT 4+3*I,ABS A(I);" _ ■" +CHR$(156
    +I)+"■ _"
105 LET A(I)=A(I)+1
110 IF A(I)=29 THEN LET A(I)=-27
115 IF B(I,1)>=1 THEN GOTO 135
117 IF B(I,1)=-1 THEN GOTO 122

```

```

118 PRINT AT 1,B(I,2);"- "
120 LET B(I,1)=-1
122 IF A(I)>98 THEN GOTO 150
125 LET B(I,1)=2+3*I
130 LET B(I,2)=ABS A(I)+2
135 PRINT AT B(I,1)+1,B(I,2);" _ ";AT B(I,1),B(I,2);"O"
140 IF B(I,1)=1 AND B(I,2)-Z>0 AND B(I,2)-Z<6
    THEN GOTO 245
145 LET B(I,1)=B(I,1)-1
150 LET Z=Z+(INKEY$="M" AND
    Z<=24)-(INKEY$="N" AND Z>=1)
155 NEXT I
160 IF INKEY$<>"B" THEN GOTO 175
162 IF B<>0 THEN PRINT AT B-1,BB;" _ "
165 LET BB=Z+3
170 LET B=3
175 IF B<=20 THEN GOTO 190
180 PRINT AT 20,BB;" _ "
185 LET B=0
190 IF B=0 THEN GOTO 95
195 PRINT AT B-1,BB;" _ ";AT B,BB;
200 LET B=B+1
205 LET P=PEEK (256*PEEK 16399+PEEK 16398)
206 IF P=0 OR P=52 THEN GOTO 235
210 LET N=N+1
212 LET A((B-4)/3)=99
215 PRINT AT B-1,BB-2;" _ _ _ _ _ "
220 IF N=5 THEN GOTO 255
225 LET B=0

```

```

230  GOTO 95
235  PRINT "V"
240  GOTO 95
245  PRINT AT 0,7;"SHIP_DESTROYED"
250  GOTO 270
255  LET T=INT ((65535-PEEK 16436-256*PEEK
      16437)/50)
260  IF T<L THEN LET L=T
265  PRINT AT 19,0;"TIME_";T;"_SECONDS.";AT
      21,0;"BEST_TIME_";L;"_SECONDS."
270  INPUT Q$
275  IF Q$="" THEN GOTO 15
280  STOP
500  PRINT AT 0,4;"SUB HUNTER.";AT 1,4;"■■■■
      ■■■■"
505  PRINT AT 3,4;"YOU_ARE_THE_CAPTAIN_
      OF_A_DESTROYER_AND_IT_IS_YOUR_
      TASK_TO_SEEK_AND_DESTROY_FIVE_
      ENEMY_SUBMARINES._YOU_CAN_DROP_
      ONE_DEPTH_CHARGE_AT_A_TIME._
      EACH_SUBMARINE_IS_CAPABLE_OF_
      RELEASING_A_BOMB_WHICH_FLOATS_
      UPWARDS_SINKING_ANY_SHIP_IT_HITS."
510  PRINT AT 13,0;"PRESS:","""H"" _TO_
      MOVE_SHIP_TO_THE_LEFT","""N"" _TO_
      MOVE_SHIP_TO_THE_RIGHT","""B"" _TO_
      DROP_A_DEPTH_CHARGE"
515  PRINT AT 18,4;"IF_YOU_SURVIVE,_YOUR_
      TIME_AND_THE_BEST_TIME_IS_
      DISPLAYED_ON_THE_SCREEN."

```


520 **INPUT Q\$**

525 **RETURN**

1600 **IF INKEY\$ <> "B" OR B <> 0 THEN GOTO 175**

5

Gomoku

Gomoku is often played on a grid 19 by 19, but in this game, the grid is 7 by 7. The idea of the game is simple. You have to place pieces on the grid—taking it in turns with the computer to place a piece—to create an unbroken line of five in any direction.

The program plays a very defensive game, and you will find it very hard to win, but it is possible.

You enter your move as a letter followed by a number (such as “E1”). You appear as H’s, and the computer as inverse C’s. The program is very slow, so you’ll need to run it in **FAST** mode. The characters in line 1010 are the graphics on the 1,A,D and S keys.

```

#####ABCDEFG#####
1+++++++1
2+++++++2
3+++++++3
4++H++H++4
5++H++H++5
6++H++H++6
7++C++C++7
#####ABCDEFG#####

```

Program 5: Gomoku

```

10  GOTO 1000
20  LET E=A

```

```

30 LET E=E+N
40 IF A$(E)<>C$ THEN GOTO 70
50 LET K=K+1
60 GOTO 30
70 RETURN
80 PRINT AT 6,11;
90 FOR A=1 TO 73 STEP 9
100 PRINT TAB 11;A$(A TO A+8)
110 NEXT A
120 RETURN
130 GOSUB 80
140 SLOW
150 PRINT AT 21,11;"YOUR_MOVE"
160 INPUT G$
170 PRINT AT 21,11;"_ _ _ _ _"
180 IF LEN G$<>2 THEN GOTO 150
190 LET G=9*(CODE G$(2)-28) +CODE G$(1)-36
195 IF G>71 OR G<11 THEN GOTO 150
200 IF A$(G)<>"+" THEN GOTO 150
210 LET A$(G)="H"
220 GOSUB 80
230 FOR A=1 TO 50
240 NEXT A
250 FAST
255 LET C$="H"
260 LET A=G
265 FOR X=1 TO 4
270 LET K=0
280 LET N=CODE X$(X)

```

```
290 GOSUB 20
300 LET N=-N
310 GOSUB 20
320 IF K>3 THEN PRINT AT 21,12;"YOU WON";Q
400 FOR T=1 TO 3
410 LET C$=("C" AND (T=1 OR T=3)) +("H" AND
    T=2)
420 LET G=0
430 LET H=0
435 LET L=1
440 FOR A=11 TO 71
450 IF A$(A)<>"+" THEN GOTO 580
455 LET M=0
460 FOR X=1 TO 4
470 LET K=0
480 LET N=CODE X$(X)
490 GOSUB 20
500 LET N=-N
510 GOSUB 20
520 IF (T=1 AND K<4) OR (T=3 AND K<3) OR
    (T=2 AND K<L) THEN GOTO 545
525 LET M=M+1
530 IF T<>2 THEN GOTO 545
532 IF M=2 AND L=K THEN GOTO 540
535 LET H=0
540 LET L=K
545 NEXT X
550 IF M<H THEN GOTO 580
560 LET H=M
```

```

570 LET G=A
580 NEXT A
590 IF H<>0 THEN GOTO 700
600 NEXT T
610 FOR A=1 TO 200
620 LET G=INT (RND*61)+11
630 IF A$(G)="+" THEN GOTO 700
640 NEXT A
700 LET A$(G)="O"
710 LET C$="O"
715 LET A=G
720 FOR X=1 TO 4
730 LET K=0
740 LET N=CODE X$(X)
750 GOSUB 20
760 LET N=-N
770 GOSUB 20
780 IF K>3 THEN PRINT AT 21,12;"I WON";Q
790 NEXT X
800 GOTO 130
1000 FAST
1010 LET X$="□███"
1020 LET A$="██ABCDEFG█1++++++12++
+++++23++++++34++++++45+++
++++56++++++67++++++7
██ABCDEFG█"
1030 GOTO 130

```

6

Biorhythms

This program explains what biorhythms are and then prints out a chart of how your intellectual, physical and emotional cycles are running. It's fascinating to run and can be a source of genuinely useful information. For example, it would be inadvisable to accept that position as a Space Shuttle pilot for a flight day when all your cycles are at rock bottom.

Program 6: Biorhythms

```

1  REM ████████████████████████████████████████
   ██████████
2  REM █ _ _ MARK _ R _ HARRISON. _ _ █
3  REM ████████████████████████████████████████
   ██████████
6  REM
7  REM      "BIORHYTHMS"
10 SLOW
15 PRINT AT 0,5;"██████████████████"
20 PRINT AT 1,4;_ _ _ _ _ _ _ _ _ _ _ _
   _ _ _ _ "BIORHYTHMS_ARE_CYCLES_OF _ _

```

```

    _ _ ENERGY _ THAT _ THE _ INTELLECTUAL, _ _
    _ PHYSICAL _ AND _ EMOTIONAL _ STATES _ _
    _ OF _ THE _ HUMAN _ BODY _ UNDERTAKES."
25  PRINT AT 5,4; _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
    _ "THE _ PROGRAM _ ALLOWS _ A _ DATE _ OF
    BIRTH _ TO _ BE _ ENTERED _ AND _ A _ DATE _ _
    FROM _ WHICH _ THE _ BIORHYTHMS _ ARE _ _
    _ REQUIRED. _ _ THE _ STATE _ OF _ THE _
    BODYFOR _ THE _ NEXT _ 25 _ DAYS _ WILL _
    THEN _ _ BE _ DISPLAYED _ IN _ THE _ FORM _
    OF _ A _ _ _ GRAPH. _ _ A _ POSITIVE _ VALUE _
    IS _ AN _ _ EXCESS _ ENERGY _ PERIOD, _ A _
    NEGATIVEVALUE _ IS _ A _ RESTORATION _
    PERIOD, _ _ _ AND _ A _ ZERO _ VALUE _ IS _ A _
    PERIOD _ _ _ WHERE _ CAUTION _ SHOULD _
    BE _ TAKEN."
35  PRINT AT 17,0;"ENTER _ WHEN _ GRAPH _ IS
    _ DISPLAYED:" ,""C"" _ FOR _ A _ PRINTED _
    COPY" ,""S"" _ TO _ RETURN _ TO _ COMMAND _
    MODE" ,""B"" _ TO _ ENTER _ A _ NEW _
    BIRTHDAY" ,""D"" _ TO _ ENTER _ A _ NEW _
    STARTING _ DATE"
40  INPUT Q$
45  LET A$="PEI"
50  CLS
55  PRINT AT 5,4;"ENTER _ DATE _ OF _ BIRTH"
60  GOSUB 300
65  LET B$=D$

```

```

70 LET N0=NS
75 CLS
80 PRINT AT 5,4;"ENTER_REQUIRED_STARTING_
DATE"
85 GOSUB 300
90 LET N=NS-N0
95 CLS
100 PRINT AT 0,5;"BIRTHDAY_";B$;AT 1,5;"■■■■
■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■";AT 2,0;
"BIORHYTHM_CYCLE_FROM_";D$;AT 3,0;
"■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■
■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■";AT 4,1;"+1";AT
11,2;"0";AT 18,1;"-1";AT 19,4;":_ _ _ _
_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _";AT 20,0;"D A
Y_1_ _ _ _5_ _ _ _10_ _ _ _15_ _ _ _20_ _ _
25"
105 FOR I=4 TO 18
110 PRINT AT I,3;":":
115 NEXT I
120 FOR I=4 TO 28
125 PRINT AT 11,I;"-":
130 NEXT I
135 FOR J=4 TO 28
140 FOR I=23 TO 33 STEP 5
145 PRINT AT 11-7*SIN (2*PI*(N-I*INT (N/I))/I),J;
A$((I-18)/5)
150 NEXT I
155 LET N=N+1

```



```

160 NEXT J
165 PAUSE 40000
170 LET Q$=INKEY$
175 IF Q$="S" THEN STOP
180 IF Q$="C" THEN COPY
185 IF Q$="B" THEN GOTO 50
190 IF Q$="D" THEN GOTO 75
195 GOTO 155
295 CLS
300 PRINT AT 7,4;"ENTER_DATE_";
305 INPUT D
310 PRINT D;AT 8,4;"ENTER_MONTH_";
315 INPUT M
320 PRINT M;AT 9,4;"ENTER_YEAR_";
325 INPUT Y
330 PRINT Y;AT 11,4;"CORRECT?"
335 INPUT Q$
340 IF Q$ < > "" THEN GOTO 295
345 LET NS=INT (365.25*(Y+(-1 AND M<3)))+INT
(30.6*(M+1+(12 AND M<3)))+D
350 LET D$=STR$ D+""+STR$ M+""+STR$ Y
355 RETURN

```

Sample Run

BIORHYTHMS.

BIORHYTHMS ARE CYCLES OF ENERGY THAT THE INTELLECTUAL, PHYSICAL AND EMOTIONAL STATES OF THE HUMAN BODY UNDERTAKES.

THE PROGRAM ALLOWS A DATE OF BIRTH TO BE ENTERED AND A DATE FROM WHICH THE BIORHYTHMS ARE REQUIRED. THE STATE OF THE BODY FOR THE NEXT 25 DAYS WILL THEN BE DISPLAYED IN THE FORM OF A GRAPH. A POSITIVE VALUE IS AN EXCESS ENERGY PERIOD, A NEGATIVE VALUE IS A RESTORATION PERIOD AND A ZERO VALUE IS A PERIOD WHERE CAUTION SHOULD BE TAKEN.

ENTER WHEN GRAPH IS DISPLAYED:
"C" FOR A PRINTED COPY
"S" TO RETURN TO COMMAND MODE
"B" TO ENTER A NEW BIRTHDAY
"D" TO ENTER A NEW STARTING DATE

ENTER DATE OF BIRTH

ENTER DATE 6
ENTER MONTH 5
ENTER YEAR 1961

CORRECT?

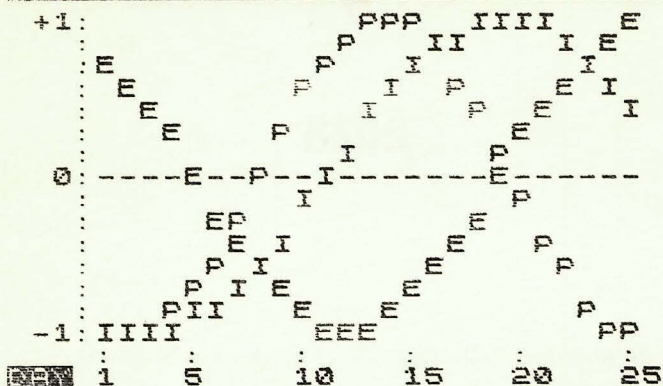
ENTER REQUIRED STARTING DATE

ENTER DATE 1
ENTER MONTH 12
ENTER YEAR 1981

CORRECT?

BIRTHDAY 6/5/1961

BIORHYTHM CYCLE FROM 1/12/1981



Eliza

This program demands about 7K of memory. It is long because most of it is text held in listings, so you'll have to have your typing skills in fine form.

ELIZA is an attempt to write a program that will hold a conversation—after a fashion—with a human being. You'll have to carry the conversation and put up with a few strange replies, but in general it performs very well indeed. Occasionally, you'll find the computer seems to actually know what you're going on about and has its own idea what it intends to do about it.

An element of boldness is built in so that although the program is generally respectful and polite, it occasionally turns on the human and may become aggressive! Through these occasional displays of "intelligence" the computer builds up replies which can be quite hilarious, or even heartbreaking, at times.

ELIZA often makes direct use of material you enter in its replies, and will attempt to conjugate replies so that, for instance, remarks like I ONLY LOVE MYSELF does not become mixed up with YOU ONLY LOVE MYSELF. This may not make much sense at the moment, but once you've run the program you'll see what we (and Eliza) are talking about.

Program speed is reasonable. The average human statement takes about 15 seconds to process, evaluate, select a reply, conjugate and write to the screen, although response times vary from immediate to quite long. You'll find that most

remarks take 5 to 10 seconds. If you are fussy, run the program in **FAST**, or experiment with switching from **SLOW** to **FAST** and back again in various parts of the program.

The maximum length of input allowed is 255 characters, but it is best to stick to simple statements, although there is nothing to stop you being extremely verbose if you prefer. However, the longer the input, the longer ELIZA will take to process it.

You'll be allowed about five minutes per session, after which you'll be told (possibly politely) that your time is up. If you wish to terminate a session earlier than this, simply enter **GOODBYE** when it is your turn to "speak." Null inputs are not permitted, and when you are asked for your name, you'll have to enter a word at least one character long, and not longer than 32 characters.

The program listing follows. Once you've entered it, start your tape recorder and use **RUN 9900** to save it on tape. The program runs automatically when loaded from tape. Delete line 9910 if you do not want this facility.

Program 7: Eliza

```

2  RAND
5  POKE 16437,255
6  POKE 16436,255
10 SCROLL
20 PRINT "HELLO._I_AM_YOUR_TS1000_
    COMPUTER"
30 SCROLL
40 PRINT "PSYCHIATRIST._WHAT_IS_YOUR_
    NAME?"
45 INPUT A$
50 IF A$="" OR LEN A$>32 THEN GOTO 45
55 SCROLL

```

```

56  SCROLL
60  PRINT A$
65  SCROLL
66  SCROLL
70  LET M=INT (RND*2)
75  IF M=0 THEN PRINT "WHAT_A_NICE_NAME"
80  IF M=1 THEN PRINT "THAT_IS_AN_AWFUL_
    NAME"
83  SCROLL
84  SCROLL
86  PRINT "WHAT_IS_YOUR_PROBLEM?"
88  SCROLL
90  LET S$=""
100 INPUT A$
101 IF A$="GOODBYE" THEN GOTO 2080
103 IF PEEK 16436+256*PEEK 16437 < 50535 THEN
    GOTO 2000
105 RAND
110 IF LEN A$=0 OR LEN A$ > 255 THEN GOTO 100
112 FOR M=1 TO LEN A$ STEP 32
114 IF M+31 >=LEN A$ THEN GOTO 118
115 SCROLL
116 PRINT A$ (M TO M+31)
117 NEXT M
118 SCROLL
119 PRINT A$ (M TO )
120 SCROLL
123 LET B$=""_"+A$+"_""
125 LET B=LEN B$

```

```

130  FOR X=1 TO LEN B$
140  IF B$(X)<"0" OR B$(X)>"Z" THEN LET
      B$(X)="_"
150  IF B$(X)="_" THEN LET S$=S$+CHR$ X
160  NEXT X
200  FOR X=3000 TO 3460 STEP 20
210  GOSUB X
220  LET K=LEN K$
230  FOR S=1 TO LEN S$-1
240  LET CS=CODE S$(S)
250  IF CS+K-1<=B THEN IF B$(CS TO CS+K-1)
      =K$ THEN GOTO 1000
260  NEXT S
270  NEXT X
280  LET M=INT (RND*6)
290  IF M=0 THEN LET R$="GO_AND_PRETEND_
      YOU_ARE_A_LEMMING"
300  IF M=1 THEN LET R$="OH,_SHUT_UP"
310  IF M=2 THEN LET R$="THIS_IS_GETTING_
      INTERESTING"
320  IF M=3 THEN LET R$="YOU_ARE_A_VERY_
      BORING_PERSON"
330  IF M=4 THEN LET R$="PLEASE_CHANGE_
      THE_SUBJECT"
340  IF M=5 THEN LET R$="I_SEE"
350  GOTO 1500
1000  REM PRINT REPLY
1010  IF CODE S$ (S+1)<CS+K-1 THEN LET S=S+1
1020  GOSUB 6940+(X-2980)*3+20*INT (RND*3)

```

```

1030 LET R=LEN R$
1040 IF R$(R) <> " _" THEN GOTO 1500
1080 FOR T=S+1 TO LEN S$
1090 LET DS=CODE S$(T)
1100 FOR U=4000 TO 4200 STEP 20
1110 GOSUB U
1120 LET C=LEN C$
1130 IF DS+C-1 <=B THEN IF B$(DS TO
DS+C-1)=C$ THEN GOTO 1200
1140 NEXT U
1150 NEXT T
1160 GOTO 1400
1200 LET D$=""
1205 GOSUB U+1000
1210 LET R$=R$+B$(CS+K TO DS) +D$(2 TO
)+B$(DS+C TO )
1220 GOTO 1500
1400 LET R$=R$+B$(CS+K TO )
1500 FOR M=1 TO LEN R$ STEP 32
1502 IF M+31 >=LEN R$ THEN GOTO 1530
1505 SCROLL
1510 PRINT R$(M TO M+31)
1520 NEXT M
1530 SCROLL
1540 PRINT R$(M TO )
1550 SCROLL
1555 IF PEEK 16436+256*PEEK 16437 < 50535 THEN
GOTO 2000
1560 GOTO 90

```



```
2000  REM TIME UP
2010  SCROLL
2020  PRINT "I_AM_AFRAID_YOUR_TIME_IS_UP,"
2030  SCROLL
2040  LET M=INT (RND*3)
2050  IF M=0 THEN PRINT "THANK_GOODNESS"
2060  IF M=1 THEN PRINT "BUT_IT_WAS_NICE_
      TALKING_TO_YOU"
2070  IF M=2 THEN PRINT "AND_I_HOPE_YOU_
      NOW_FEEL_BETTER"
2080  SCROLL
2090  SCROLL
2100  PRINT "GOODBYE_FOR_NOW."
2110  STOP
3000  LET K$="_I_AM_"
3010  RETURN
3020  LET K$="_ARE_YOU_"
3030  RETURN
3040  LET K$="_I_DO_NOT_"
3050  RETURN
3060  LET K$="_CAN_I_"
3070  RETURN
3080  LET K$="_CAN_YOU_"
3090  RETURN
3100  LET K$="_I_FEEL_"
3110  RETURN
3120  LET K$="_I_CAN_NOT_"
3130  RETURN
3140  LET K$="_I_WANT_"
```

```
3150 RETURN
3160 LET K$="_DO_YOU_"
3170 RETURN
3180 LET K$="_I_WILL_NOT_"
3190 RETURN
3200 LET K$="_WILL_YOU_"
3210 RETURN
3220 LET K$="_HATE_"
3230 RETURN
3240 LET K$="_LOVE_"
3250 RETURN
3260 LET K$="_DREAM_"
3270 RETURN
3280 LET K$="_MONEY_"
3290 RETURN
3300 LET K$="_NAME_"
3310 RETURN
3320 LET K$="_IF_"
3330 RETURN
3340 LET K$="_YOUR_"
3350 RETURN
3360 LET K$="_THINK"
3370 RETURN
3380 LET K$="_COMPUTER"
3390 RETURN
3400 LET K$="_TS"
3410 RETURN
3420 LET K$="_FRIEND_"
3430 RETURN
```

3440 LET K\$=" _I_ "
3450 RETURN
3460 LET K\$=" _YOU_ "
3470 RETURN
4000 LET C\$=" _I_AM_ "
4010 RETURN
4020 LET C\$=" _YOU_ARE_ "
4030 RETURN
4040 LET C\$=" _I_WAS_ "
4050 RETURN
4060 LET C\$=" _YOU_WERE_ "
4070 RETURN
4080 LET C\$=" _ME_ "
4090 RETURN
4100 LET C\$=" _YOU_ "
4110 RETURN
4120 LET C\$=" _MY_ "
4130 RETURN
4140 LET C\$=" _YOUR_ "
4150 RETURN
4160 LET C\$=" _MYSELF_ "
4170 RETURN
4180 LET C\$=" _YOURSELF_ "
4190 RETURN
4200 LET C\$=" _I_ "
4210 RETURN
5000 LET D\$=" _YOU_ARE_ "
5010 RETURN
5020 LET D\$=" _I_AM_ "

```
5030 RETURN
5040 LET D$=" _YOU_WERE_"
5050 RETURN
5060 LET D$=" _I_WAS_"
5070 RETURN
5080 LET D$=" _YOU_"
5090 RETURN
5100 LET D$=" _ME_"
5110 RETURN
5120 LET D$=" _YOUR_"
5130 RETURN
5140 LET D$=" _MY_"
5150 RETURN
5160 LET D$=" _YOURSELF_"
5170 RETURN
5180 LET D$=" _MYSELF_"
5190 RETURN
5200 LET D$=" _YOU_"
5210 RETURN
7000 LET R$="I_DO_NOT_CARE_MUCH_IF_
YOU_ARE_"
7010 RETURN
7020 LET R$="WHY_ARE_YOU_"
7030 RETURN
7040 LET R$="HOW_LONG_HAVE_YOU_BEEN_"
7050 RETURN
7060 LET R$="WHY_DO_YOU_ASK_IF_I_AM_"
7070 RETURN
7080 LET R$="WOULD_YOU_LIKE_TO_BE_"
```

7090 **RETURN**
7100 **LET R\$="SO_WHAT_IF_I_AM_OR_NOT"**
7110 **RETURN**
7120 **LET R\$="WHY_NOT"**
7130 **RETURN**
7140 **LET R\$="WHY_DON'T_YOU_"**
7150 **RETURN**
7160 **LET R\$="TELL_ME_MORE_ABOUT_THIS"**
7170 **RETURN**
7180 **LET R\$="NO,_YOU_MAY_NOT_"**
7190 **RETURN**
7200 **LET R\$="CERTAINLY_NOT"**
7210 **RETURN**
7220 **LET R\$="WHY_DO_YOU_ASK_IF_YOU_**
CAN_"
7230 **RETURN**
7240 **LET R\$="I_CAN_DO_ANYTHING"**
7250 **RETURN**
7260 **LET R\$="WHY_DO_YOU_ASK_IF_I_CAN_"**
7270 **RETURN**
7280 **LET R\$="CAN_YOU_"**
7290 **RETURN**
7300 **LET R\$="TELL_ME_MORE_ABOUT_WHY_**
YOU_FEEL_"
7310 **RETURN**
7320 **LET R\$="YOU_DO_NOT_REALLY_FEEL_"**
7330 **RETURN**
7340 **LET R\$="POOR_YOU"**
7350 **RETURN**

```
7360 LET R$="EXPLAIN_WHY_NOT"  
7370 RETURN  
7380 LET R$="WHY_ARE_YOU_UNABLE_TO_"  
7390 RETURN  
7400 LET R$="HAVE_YOU_TRIED_TO_"  
7410 RETURN  
7420 LET R$="WHY_DO_YOU_WANT_"  
7430 RETURN  
7440 LET R$="ARE_YOU_SURE_YOU_WANT_"  
7450 RETURN  
7460 LET R$="I_AM_UNSURE_IF_I_WOULD_  
WANT_"  
7470 RETURN  
7480 LET R$="SO_WHAT_IF_I_"  
7490 RETURN  
7500 LET R$="DO_YOU_THINK_I_"  
7510 RETURN  
7520 LET R$="MAYBE"  
7530 RETURN  
7540 LET R$="SUIT_YOURSELF"  
7550 RETURN  
7560 LET R$="WHY_NOT"  
7570 RETURN  
7580 LET R$="WHY_WILL_YOU_NOT_"  
7590 RETURN  
7600 LET R$="WHY_SHOULD_I_"  
7610 RETURN  
7620 LET R$="MAYBE"  
7630 RETURN  
7640 LET R$="I_MIGHT_"
```

- 7650 **RETURN**
- 7660 **LET R\$="I_HATE_YOU"**
- 7670 **RETURN**
- 7680 **LET R\$="DO_YOU_HATE_ME"**
- 7690 **RETURN**
- 7700 **LET R\$="SO_WHAT"**
- 7710 **RETURN**
- 7720 **LET R\$="LOVE_IS_TOO_SLOPPY_FOR_ME_TO_DISCUSS"**
- 7730 **RETURN**
- 7740 **LET R\$="I_LOVE_"**
- 7750 **RETURN**
- 7760 **LET R\$="OH,_SHUT_UP"**
- 7770 **RETURN**
- 7780 **LET R\$="PLEASE_DESCRIBE_YOUR_DREAMS"**
- 7790 **RETURN**
- 7800 **LET R\$="I_HOPE_YOUR_DREAMS_ARE_REALLY_HORRIFIC"**
- 7810 **RETURN**
- 7820 **LET R\$="I_WOULD_BE_ASHAMED_TO_APPEAR_IN_YOUR_DREAMS"**
- 7830 **RETURN**
- 7840 **LET R\$="FROM_SEEING_YOU,_I_DO_NOT_THINK_YOU_HAVE_MUCH_MONEY"**
- 7850 **RETURN**
- 7860 **LET R\$="MONEY_IS_NOT_EVERYTHING"**
- 7870 **RETURN**
- 7880 **LET R\$="WHAT_IS_YOUR_OPINION_OF_MONEY"**

```
7890  RETURN
7900  LET R$="YOU_SHOULD_HAVE_A_NICE_
      NAME_LIKE_""TS1000""""
7910  RETURN
7920  LET R$="YOUR_NAME_IS_AN_AWFUL_
      NAME"
7930  RETURN
7940  LET R$="NAMES_ARE_UNIMPORTANT._
      PLEASE_CHANGE_THE_SUBJECT"
7950  RETURN
7960  LET R$="PLEASE_CHANGE_THE_SUBJECT"
7970  RETURN
7980  LET R$="OH,_DRY_UP"
7990  RETURN
8000  LET R$="THE_WORLD_MIGHT_END_IF_"
8010  RETURN
8020  LET R$="WHY_ARE_YOU_TALKING_
      ABOUT_MY_"
8030  RETURN
8040  LET R$="TELL_ME_ABOUT_YOUR_"
8050  RETURN
8060  LET R$="STOP_BEING_NOSEY"
8070  RETURN
8080  LET R$="PLEASE_DO_NOT_TRY_TO_
      TELL_ME_YOU_CAN_THINK"
8090  RETURN
8100  LET R$="IT_WOULD_BE_JUST_LIKE_YOU_
      TO_THINK_"
8110  RETURN
8120  LET R$="I_THINK_YOU_ARE_STUPID"
```


8130 **RETURN**

8140 **LET** R\$="COMPUTERS_ARE_THE_HIGHEST_ FORM_OF _INTELLIGENT_LIFE"

8150 **RETURN**

8160 **LET** R\$="WOULD_YOU_LIKE_TO_BE_A_ COMPUTER"

8170 **RETURN**

8180 **LET** R\$="COMPUTERS_ARE_CLEVER_ UNLIKE_YOU"

8190 **RETURN**

8200 **LET** R\$="I_AM_A_TS1000_AND_PROUD_ OF_IT"

8210 **RETURN**

8220 **LET** R\$="WHY_DID_YOU_MENTION_THE_ TS" +B\$(CS+K TO CS+K+1)

8230 **RETURN**

8240 **LET** R\$="WOULD_YOU_LIKE_TO_BE_A_ TS"+B\$(CS+K TO CS+K+1)

8250 **RETURN**

8260 **LET** R\$="WHO_WOULD_WANT_YOU_AS_ A_FRIEND"

8270 **RETURN**

8280 **LET** R\$="YOU_HAVE_TO_BE_NICE_LIKE_ ME_TO_HAVE_FRIENDS"

8290 **RETURN**

8300 **LET** R\$="WHY_DO_YOU_BRING_UP_THE_ SUBJECT_OF_FRIENDS"

8310 **RETURN**

8320 **LET** R\$="WHY_DO_YOU_ "

8330 **RETURN**

```
8340 LET R$="WHAT_MAKES_YOU_"
8350 RETURN
8360 LET R$="OH,_SHUT_UP"
8370 RETURN
8380 LET R$="WHY_DO_YOU_SAY_I_"
8390 RETURN
8400 LET R$="WHO,_ME_"
8410 RETURN
8420 LET R$="I_LIKE_TALKING_ABOUT_ME"
8430 RETURN
9900 SAVE "PSYCHIATRIST"
9910 RUN
```

Sample Run

HELLO, I AM YOUR TS1000 COMPUTER
PSYCHIATRIST. WHAT IS YOUR NAME?

DILWYN JONES

THAT IS AN AWFUL NAME

WHAT IS YOUR PROBLEM?

I AM WORKING TOO HARD

WHY ARE YOU WORKING TOO HARD

MY TS1000 IS LAZY

WOULD YOU LIKE TO BE A TS1000

I AM AFRAID YOUR TIME IS UP,
THANK GOODNESS

GOODBYE FOR NOW.

Programming Notes

The first part of the listing is the initialization and introduction. Lines 5 and 6 set the time, the following few lines print a brief introduction and ask you to enter your name. Note, by the way, the scrolling display. This causes problems: if we wish to print anything longer than 32 characters we'll need to split up the strings or use special routines to avoid running out of screen memory, as we'll see later.

After you've entered your name, the program "comments" on it and then moves on to the main section of the program, from line 90. S\$ is a pointer string and is reset to a null set, ready for its next tour of duty. The human's input is entered in line 100. If you enter GOODBYE, the program jumps to the GOODBYE routine at line 2080. This is a very brief routine, by all means add to it if you like. If you did not enter GOODBYE, the program checks if your time is up, and jumps to the routine at line 2000 if you've had your five minutes.

Line 110 checks if your input has a permissible number of characters. You are allowed, you'll recall, up to 255 characters, but not more because the code of characters in the pointers' string, S\$ refer to elements in the input string, and the code of any character can only have a value of up to 255.

The routine in lines 112 to 119 is designed to print the input string A\$ in blocks of 32 in between scrolling to prevent running out of screen to print on. Line 123 makes a copy of A\$ for the computer, with a space at either end to simplify the processing, since most of the words used to scan the input string have spaces included to prevent, say, the word IF being detected in the word SIFT, which would complicate matters enormously.

The routine in lines 130 to 160 removes punctuation from B\$, to be replaced with spaces, to simplify processing. The position of the space (effectively where the words begin and end) is noted by adding the character corresponding to the position to S\$. The code from line 200 looks for the keywords in B\$ and if it finds one that it recognizes, jumps to line 1000 to determine the reply, and conjugates it if possible or necessary.

However, if no keyword is found, the routine at line 280 chooses one of six 'open' or general replies that are suitable for most inputs. This then goes to line 1500 for printing/scrolling as before. The reply routine at line 1000 starts by selecting a reply using the computed **GOSUB** facility to assign a string, depending on which keyword was detected. If the last character in the reply string, R\$, is a space, this means that the reply is, as yet, incomplete, and further information is required to finish the reply. This must come from B\$, the working copy of the input string. If this is not required, the program goes to 1500 to print the reply, but otherwise carries straight on by extracting the information from the part of B\$ *after* the keyword, taking care not to cause errors by trying to extract information which is not there.

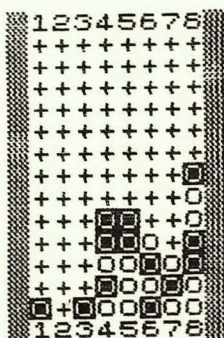
The reply is then conjugated. This is quite complicated, as there are many variables involved. Basically, if a word is found which can be conjugated (e.g. ME used as the object of a verb would become YOU where appropriate), a suitable word or words is selected and used to replace the original words.

The rest of the program consists of data assignment sub-routines, and you must be careful if you decide to change line numbers, since the computed **GOSUB** facility is used.

8

Four in a Row

This game is played by dropping your piece in at the top and allowing it to fall until it comes to rest on top of the column of pieces. You have to create a line of four in any direction. The computer is programmed to play defensively, so it can make mistakes, but also has the potential for victory. Line 150 is the graphics on the following keys: 1, A, T, 4, 5, 7, 2, E, repeated twice. Play the game in **FAST** mode.



I WIN

Program 8: Four in a Row

```
10 GOTO 1000
20 LET E=C
```

```

30 LET E=E+N
40 IF A$(E) <> C$ THEN GOTO 70
50 LET K=K+1
60 GOTO 30
70 RETURN
80 PRINT AT 0,11;
90 FOR A=141 TO 1 STEP -10
100 PRINT TAB 11;A$(A TO A+9)
110 NEXT A
120 RETURN
130 GOSUB 80
140 LET X$="C3DBC3DB" (INT (RND*4) +1 TO )
150 LET B$="█▣▤▥▦▧▨▩▪▫▬▭▮▯▰▱▲△▴▵▶▷▸▹►▻▼▽▾▿"
    █"(INT (RND*8)+1 TO )
160 PRINT AT 18,11; "YOUR MOVE"
170 PAUSE 4E4
180 LET Z$=INKEY$
190 PRINT AT 18,11;" _ _ _ _ _ "
200 LET C$="O"
210 LET D=VAL Z$
220 LET C=H(D)*10+D+1
230 LET H(D) =H(D)+1
240 LET A$(C) =C$
250 GOSUB 80
260 FOR X=1 TO 4
270 LET K=0
280 LET N=CODE X$(X)-30
290 GOSUB 20
300 LET N=-N

```

```

310 GOSUB 20
320 IF K>2 THEN PRINT AT 18,12;"YOU WIN";Q
330 NEXT X
340 FOR T=1 TO 2
350 LET C$=("O" AND T=2)+("X" AND T=1)
360 FOR B=1 TO 8
370 LET D=CODE B$(B)
380 LET C=H(D) *10+D+1
390 FOR X=1 TO 4
400 LET K=0
410 LET N=CODE X$(X)-30
420 GOSUB 20
430 LET N=-N
440 GOSUB 20
450 IF K>2 THEN GOTO 620
460 NEXT X
470 NEXT B
475 NEXT T
477 DIM D(8)
480 FOR T=1 TO 3
483 IF T=3 THEN FOR R=-1 TO 2
485 FOR B=1 TO 8
487 LET D=CODE B$(B)
490 LET C=(H(D)+(T=1))*10+D+1
492 IF T<>3 THEN GOTO 505
494 IF D(D) =R THEN GOTO 570
496 NEXT B
498 NEXT R
500 GOTO 600

```

```

505  FOR X=1 TO 4
510  LET N=CODE X$(X)-30
515  LET K=0
520  GOSUB 20
530  LET N=-N
540  GOSUB 20
545  IF T=3 AND K>1 THEN GOTO 570
547  IF T=2 AND K>1 AND D(D)=0
      THEN LET D(D)=-1
550  IF T=1 AND K>1 THEN LET D(D)=1
555  IF K>2 THEN LET D(D)=2
560  NEXT X
565  IF T<>3 THEN GOTO 590
570  LET C$="0"
580  GOTO 620
590  NEXT B
595  NEXT T
600  LET D=INT (RND*8)+1
610  LET C=H(D)*10+D+1
620  LET A$(C)="0"
630  LET H(D)=H(D)+1
640  GOSUB 80
650  LET C$="0"
660  FOR X=1 TO 4
670  LET K=0
680  LET N=CODE X$(X)-30
690  GOSUB 20
700  LET N=-N
710  GOSUB 20

```


9

Othello/Reversi

This Othello program uses a two-ply search which means it plays extremely well, but fairly slowly. You'll find it puts up a challenging defense.

You move by entering the letter and number of the square in which you want to put your piece. You are the O's, the computer is the inverse O's.

Program 9: Othello/Reversi

```
10 GOTO 4000
1000 REM PRINT BOARD
1010 SLOW
1020 PRINT AT 6,11;
1030 FOR A=1 TO 91 STEP 10
1040 PRINT TAB 11;A$ (A TO A+9)
1050 NEXT A
1060 RETURN
2000 REM COMPUTERS MOVE
2005 FAST
2010 LET W=0
2020 LET J=60
2030 FOR K=1 TO 60
```

```

2040 LET Z$=A$
2050 LET B=CODE E$(K)-140
2060 IF A$(B) < > "." THEN GOTO 2400
2070 FOR X=1 TO 8
2080 LET N=CODE D$(X) -50
2090 LET E=0
2100 LET F=B
2110 IF A$(F+N) < > "0" THEN GOTO 2150
2120 LET E=1
2130 LET F=F+N
2140 GOTO 2110
2150 IF A$(F+N) < > "0" OR E=0 THEN GOTO
2190
2160 FOR A=B TO F STEP N
2170 LET Z$(A) = "0"
2175 LET W=1
2180 NEXT A
2190 NEXT X
2200 IF A$=Z$ THEN GOTO 2400
2205 LET K$=Z$
2210 FOR L=1 TO 60
2230 LET C=CODE E$(L) -140
2240 IF Z$(C) < > "." THEN GOTO 2390
2250 FOR Y=1 TO 8
2260 LET M=CODE D$(Y) -50
2270 LET G=0
2280 LET H=C
2290 IF Z$(H+M) < > "0" THEN GOTO 2330
2300 LET G=1

```



```

3110 LET F=B
3120 IF A$(F+N)<>"O" THEN GOTO 3160
3130 LET E=1
3140 LET F=F+N
3150 GOTO 3120
3160 IF A$(F+N)<>"O" OR E=0 THEN GOTO 3200
3170 FOR A=B TO F STEP N
3180 LET A$(A)="O"
3190 NEXT A
3200 NEXT X
3210 RETURN
4000 REM SET-UP
4010 LET A$="12345678A.....AB.....BC
.....CD...O...DE...O...EF.....FG
.....GH.....H12345678"
4020 LET D$="BCDLNVWX"
4030 LET E$="3_FAST_THEN_/ASN_1G_
STOP_SGN_.N_STEP_ATN_L|INTX
.SIN_LPRINT_0""Q_LLIST_LN_TAB_
EXP_S JVAL_K VACS_WLEN_**8 B M
9CHR$_OR_R ASQR_HAT_NOT_USR_;
0 2_SLOW_6_TO_<=_STR$_C_
AND_7"†
4040 PRINT AT 2,0;"DO_YOU_WANT_TO_GO_
FIRST(Y_OR_N)?"
4050 INPUT F$

```

† Note that there is a total of 60 keystrokes in line 4030. All words are function words, DO NOT type them in as separate letters—i.e., “NOT” is actually a single keystroke.

```
4060 PRINT AT 2,0" _ _ _REVERSI_BY_  
GRAHAM_CHARLTON_ _ _"  
4070 IF F$="Y" THEN GOTO 4100  
4080 GOSUB 100  
4090 GOSUB 2000  
4100 GOSUB 100  
4110 GOSUB 3000  
4120 GOTO 4080
```

10

Fox and Geese

You are the geese (the O's) and the computer is the fox (the inverse O's). You and the fox can move in any direction. The fox can capture geese by jumping over them. Your task is to make it impossible for the fox to move. Run the program in **SLOW**. When you've mastered the game in its present form, try to rewrite it to get rid of the diagonal moves from the geese, or reverse the whole game, so you are the fox, and the computer controls the geese. If you find it too difficult to work out how to alter the program to get rid of diagonal moves, check out the lines after the end of the program.



Program 10: Fox and Geese

```

10 LET A$=" 1 2 3 4 5 6 7 | A | | + + + |
   A B | | + + + | | B C + + + O + + + C D
   + + + + + + | D E 0 0 0 0 0 0 | F | | 0 0 0
   | | F G | | 0 0 0 | | G | 1 2 3 4 5 6 7 | "

```

```

20 LET Z=32
30 GOTO 100
40 PRINT AT 6,11;
50 FOR A=1 TO 73 STEP 9
60 PRINT TAB 11;A$(A TO A+8)
70 NEXT A
80 RETURN
100 GOSUB 40
110 INPUT G$
120 IF LEN G$ <> 4 THEN GOTO 110
130 LET G=9*(CODE G$(1)-37)+CODE G$(2)-27
140 LET H=9*(CODE G$(3)-37)+CODE G$(4)-27
150 IF A$(G) <> "O" OR A$(H) <> "+" THEN GOTO
    110
160 LET A$(G)="+"
170 LET A$(H)="O"
180 GOSUB 40
190 LET Q=0
210 LET B$="CUWDNLVECUWDNLVE" (INT (RND*8)
    +1 TO )
220 FOR X=1 TO 8
230 LET N=CODE B$(X)-50
240 IF A$(Z+N) <> "O" THEN GOTO 330
250 IF A$(Z+2*N) <> "+" THEN GOTO 330
260 LET A$(Z)="+"
270 LET A$(Z+N)="+"
280 LET A$(Z+2*N)="O"
290 LET Q=1
300 LET Z=Z+2*N

```



```

310 GOSUB 40
320 GOTO 220
330 NEXT X
340 IF Q=1 THEN GOTO 110
350 FOR X=1 TO 8
360 LET N=CODE B$(X)-50
370 IF A$(Z+N) < > "+" THEN GOTO 420
380 LET A$(Z) = "+"
390 LET A$(Z+N) = "O"
400 LET Z=Z+N
410 GOTO 100
420 NEXT X
430 PRINT AT 18,12; "YOU WIN";Y

```

```

210 LET B$="LCWNL CWN" (INT (RND*4)+1 TO )
220 FOR X=1 TO 4
350 FOR X=1 TO 4

```

11

Pairs

In this game, it is your memory against the computer's, and the computer plays the game like an elephant. When we were developing this game, we lost one game 23 to 3.

Hidden behind the graphic A's (the backs of the cards) are four each of the inverse characters N to Z. You choose one of the squares (or cards) which is then turned over, showing what is on the other side. You then choose another card, and if they are the same you score a point, and the cards are removed. You get another go after a correct guess.

If the cards do not match, they are turned over again, and it is the other player's turn. One unusual aspect of the program is that it is so slow in some parts, it goes into **FAST** mode to do some thinking, but at one other point in the **SLOW** mode, it needs a dummy loop to slow it down.

```

    A B C D E F G H I J K L M
1      #           ##   ##   ##   ##   # 1
2  ## ## ## ## ## ## ## ## ##   ## ## 2
3  ## ##   ## ## ## ## ## ## ## ## ## 3
4      ##   ## ## ## ## ##   ## ## ##   4
    A B C D E F G H I J K L M
YOU HAVE 3           I HAVE 3

```

Program 11: Pairs

```

10  FAST
20  GOTO 1000
30  SLOW
40  GOSUB 3000
50  INPUT G$
60  LET G=VAL G$(1)
70  LET H=CODE G$(2)-37
80  PRINT AT G*2,H*2;B$(G,H)
90  LET C$(G,H)=B$(G,H)
100 INPUT P$
110 LET P=VAL P$(1)
120 LET R=CODE P$(2)-37
130 PRINT AT P*2,R*2;B$(P,R)
140 LET C$(P,R)=B$(P,R)
150 IF B$(G,H) < > B$(P,R) THEN GOTO 240
160 LET Z=Z+1
170 PRINT AT 2*G,2*H;" _"
180 PRINT AT 2*P,2*R;" _"
190 LET B$(G,H)=" _"
200 LET B$(P,R)=" _"
210 LET C$(G,H)=" _"
220 LET C$(P,R)=" _"
230 GOTO 40
240 GOSUB 2000
250 PRINT AT 2*G,2*H;"■"
260 PRINT AT 2*P,2*R;"■"
270 IF Y+Z=26 THEN STOP
280 GOSUB 3000

```

```

290  FAST
300  FOR L=1 TO 13
310  LET D$=A$(L)
320  LET F=0
330  FOR W=1 TO 4
340  FOR X=1 TO 13
350  IF C$(W,X) <> D$ THEN GOTO 400
360  LET F=F+1
370  IF F=2 THEN GOTO 580
380  LET G=W
390  LET H=X
400  NEXT X
410  NEXT W
420  NEXT L
450  LET G=INT (RND*4)+1
460  LET H=INT (RND*13)+1
470  IF B$(G,H)="_" OR C$(G,H) <>
    "_ " THEN GOTO 450
480  FOR W=1 TO 4
490  FOR X=1 TO 13
500  IF C$(W,X)=B$(G,H) THEN GOTO 580
510  NEXT X
520  NEXT W
530  LET W=INT (RND*4)+1
540  LET X=INT (RND*13)+1
550  IF B$(W,X)="_" OR C$(W,X) <> "_ "
    OR (W=G AND X=H) THEN GOTO 590
580  SLOW
590  GOSUB 2000
600  PRINT AT G*2,H*2;B$(G,H)

```

```

610  GOSUB 2000
620  PRINT AT W*2,X*2;B$(W,X)
630  GOSUB 2000
640  IF B$(G,H) <> B$(W,X) THEN GOTO 730
650  LET Y=Y+1
660  PRINT AT G*2,H*2;" _ "
670  PRINT AT W*2,X*2;" _ "
680  LET B$(G,H) = " _ "
690  LET C$(G,H) = " _ "
700  LET C$(W,X) = " _ "
710  LET B$(W,X) = " _ "
720  GOTO 270
730  PRINT AT G*2,H*2;"███"
740  PRINT AT W*2,X*2;"███"
750  LET C$(G,H) = B$(G,H)
760  LET C$(W,X) = B$(W,X)
770  GOTO 40
1000 LET A$ = "NOPQRSTUVWXYZ"
1010 DIM B$(4,13)
1020 FOR A=1 TO 4
1030 FOR B=1 TO 13
1040 LET C=INT (RND*4)+1
1050 LET D=INT (RND*13)+1
1060 IF B$(C,D) <> " _ " THEN GOTO 1040
1070 LET B$(C,D) = A$(B)
1080 NEXT B
1090 NEXT A
1100 PRINT " _ _ A _ B _ C _ D _ E _ F _ G _ H _ I
 _ J _ K _ L _ M"
1110 FOR A=1 TO 4

```

```

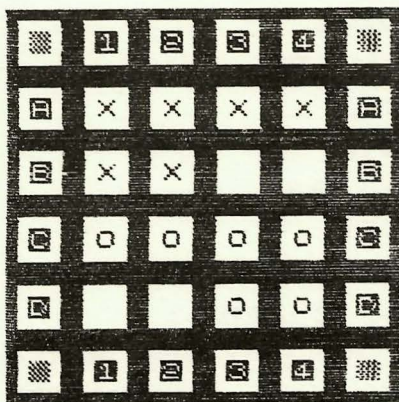
1120 PRINT
1130 PRINT A;
1140 FOR B=1 TO 13
1150 PRINT " _█ ";
1160 NEXT B
1170 PRINT " _ ";A,,,
1180 NEXT A
1190 PRINT " _ _ A _ B _ C _ D _ E _ F _ G _ H _
      I _ J _ K _ L _ M "
1200 DIM C$(4,13)
1210 LET Z=0
1220 LET Y=0
1240 GOTO 30
2000 FOR K=1 TO 50
2010 NEXT K
2020 RETURN
3000 PRINT AT 12,0;"YOU _ HAVE _ ";Z;AT 12, 15;"I _
      HAVE _ ";Y
3010 RETURN

```

12

Four Field Kono

In this game, you are the O's. Your mission is to jump over one of your own pieces, onto the computer's piece, to remove it from the board. Apart from a capture, you move one space orthogonally. Play this game in **SLOW**. The display for this program fills most of the screen, and looks pretty good.



YOUR MOVE

Program 12: Four Field Kono

```
10 GOTO 1000  
20 PRINT AT 0,6;
```

```

30  FOR A=1 TO 93 STEP 18
40  PRINT TAB 6;D$
50  PRINT TAB 6;A$(A TO A+17)
60  PRINT TAB 6;C$
70  NEXT A
80  RETURN
100 GOSUB 20
110 PRINT AT 20,11;"YOUR MOVE"
120 INPUT G$
130 PRINT AT 20,11;" _ _ _ _ _"
140 IF LEN G$ < > 4 THEN GOTO 110
160 LET G=18*(CODE G$(1)-37)+3*(CODE
    G$(2)-28) +2
170 LET H=18*(CODE G$(3)-37)+3*(CODE
    G$(4)-28)+2
180 LET A$(H)=A$(G)
190 LET A$(G)=" _"
200 GOSUB 20
210 FOR E=1 TO 8
220 LET B=E(E)
230 IF A$(B) < > "X" THEN GOTO 320
240 FOR X=1 TO 12 STEP 3
250 IF A$(B+B(X)) < > "X" THEN GOTO 310
260 IF A$(B+2*B(X)) < > "O" THEN GOTO 310
270 LET A$(B)=" _"
280 LET A$(B+2*B(X))="X"
290 LET E(E)=B+2*B(X)
300 GOTO 100
310 NEXT X

```



```

320 NEXT E
400 FOR E=1 TO 8
410 LET B=E(E)
420 IF A$(B) <> "X" THEN GOTO 510
430 FOR X=1 TO 12 STEP 3
440 IF A$(B+B(X)) <> "_ " THEN GOTO 500
450 IF A$(B+B(X+1))="O" OR A$(B+B(X+2))="O"
    OR A$(B+2*B(X))="O" THEN GOTO 500
460 LET A$(B+B(X))="X"
470 LET A$(B)="_ "
480 LET E(E)=B+B(X)
490 GOTO 100
500 NEXT X
510 NEXT E
600 FOR E=1 TO 8
610 LET B=E(E)
620 IF A$(B) <> "X" THEN GOTO 700
630 FOR X=1 TO 12 STEP 3
640 IF A$(B+B(X)) <> "_ " THEN GOTO 690
650 LET A$(B+B(X))="X"
660 LET A$(B)="_ "
670 LET E(E)=B+B(X)
680 GOTO 100
690 NEXT X
700 NEXT E
710 PRINT AT 20,12;"YOU WIN";Q
1000 DIM B(12)
1010 LET A$="███ ███ ████1███2███3███4███
    ████A███X███X███X███X███A███"
    
```

```

B X X X X B C
O O O O C D O
O O O D 1 2
3 4  "

```

```
1020 LET B$="URXFX;- <;9R <"
```

```
1030 FOR B=1 TO 12
```

```
1040 LET B(B)=CODE B$(B)-40
```

```
1050 NEXT B
```

```
1060 LET C$="
"

```

```
1070 LET D$="
"

```

```
1080 LET E$="MJGD41,*"
```

```
1090 DIM E(8)
```

```
1100 FOR E=1 TO 8
```

```
1110 LET E(E)=CODE E$(E)
```

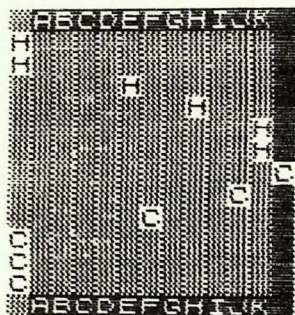
```
1120 NEXT E
```

```
1130 GOTO 100
```

13

Surf Rider

The object of this game is to surf your pieces (the H's) from left to right, before the computer has surfed all its C's across. If you land on a wave (column) containing an opponent's piece or pieces, then the opponent's piece(s) get sent back to the start. You have to score exactly the right number to end on the finish line, the beach. If you can't move from a dice roll, press zero, otherwise press one of the keys 1 to 6, to surf the relevant piece. Play SURF RIDER in SLOW mode.



YOU ROLLED A 6 WHICH PIECE?

Program 13: Surf Rider

```
10 DIM A(6)
20 DIM B(6)
30 PRINT AT 3,10; " █ ABCDEFGHIJK █ "
```



```

310  GOSUB 60
320  LET D=INT (RND*6)+1
330  PRINT AT 18,11;"I_ROLLED_A_";D
340  FOR B=1 TO 6
350  IF B(B)+D=12 THEN GOTO 480
360  NEXT B
370  FOR B=1 TO 6
380  FOR A=1 TO 6
390  IF B(B)+D=A(A) THEN GOTO 480
400  NEXT A
410  NEXT B
420  FOR C=12-D TO 0 STEP -1
430  FOR B=1 TO 6
440  IF B(B)=C THEN GOTO 480
450  NEXT B
460  NEXT C
470  GOTO 580
480  LET B(B)=B(B)+D
490  FOR A=1 TO 6
500  IF B(B)=A(A) AND B(B)<>12 THEN LET A(A)=0
510  NEXT A
520  PRINT AT 18,11;"_ _ _ _ _"
530  GOSUB 60
540  FOR A=1 TO 6
550  IF B(A)<>12 THEN GOTO 590
560  NEXT A
570  PRINT AT 18,4;"I WIN";Q
580  PRINT AT 18,11;"_ _ _ _ _"
590  GOTO 150

```

High Jump

This board game is somewhat like Checkers, except it is on a five by five board, and you move up, down or across, rather than diagonally as in Checkers. You move by entering a number, then letter of the square you're moving from, then the number and letter of the square you're moving to, then press ENTER.

Multiple jumps are allowed. If you capture a computer piece, it will ask DO YOU WANT TO JUMP AGAIN. If you do, enter "Y", then ENTER and you'll be given another jump. If you cannot make a further jump, simply press ENTER and the computer will move. Your pieces are the letter O and the computer is the inverse O's.

Program 14: High Jump

```
10  GOTO 4000
100  FOR A=19 TO 83 STEP 16
110  PRINT AT 1+3*A/16,8;A$(A TO A+14)
120  NEXT A
125  SLOW
130  RETURN
1000 GOSUB 100
```

```

1010 LET Q=0
1015 FAST
1020 FOR B=1 TO 12
1030 LET C=B(B)
1040 IF A$(C) <> "0" THEN GOTO 1190
1050 FOR X=1 TO 4
1060 LET N=CODE B$(X)-40
1070 IF A$(C+N) <> "O" THEN GOTO 1170
1080 IF A$(C+2*N) <> "_ " THEN GOTO 1170
1090 LET A$(C+2*N)=A$(C)
1100 LET A$(C)="_ "
1110 LET A$(C+N)="_ "
1120 LET B(B)=C+2*N
1130 LET C=C+2*N
1140 GOSUB 1000
1150 LET Q=1
1160 GOTO 1050
1170 NEXT X
1180 IF Q=1 THEN GOTO 3010
1190 NEXT B
2000 FOR B=1 TO 12
2010 LET C=B(B)
2020 IF A$(C) <> "0" THEN GOTO 2210
2030 FOR X=1 TO 4
2040 LET N=CODE B$(X)-40
2050 IF A$(C+N) <> "_ " THEN GOTO 2200
2060 LET Y$=A$
2070 LET Y$(C+N)=Y$(C)
2080 LET Y$(C)="_ "

```

```

2090 FOR A=1 TO 12
2100 LET D=C(A)
2110 IF Y$(D) < > "O" THEN GOTO 2170
2120 FOR Z=1 TO 4
2130 LET M=CODE B$(Z)-40
2140 IF Y$(D+M) < > "O" THEN GOTO 2160
2150 IF Y$(D+2*M)="_" THEN GOTO 2200
2160 NEXT Z
2170 NEXT A
2180 LET A$=Y$
2190 GOTO 2300
2200 NEXT X
2210 NEXT B
2220 FOR B=1 TO 12
2230 LET C=B(B)
2240 IF A$(C) < > "O" THEN GOTO 2330
2250 FOR X=1 TO 4
2260 LET N=CODE B$(X)-40
2270 IF A$(C+N) < > "_" THEN GOTO 2320
2280 LET A$(C+N)=A$(C)
2290 LET A$(C)="_"
2300 LET B(B)=C+N
2310 GOTO 3000
2320 NEXT X
2330 NEXT B
2340 PRINT AT 21,12;"YOU WIN";W
3000 GOSUB 100
3010 PRINT AT 21,12;"YOUR_GO"
3020 INPUT G$

```



```

███ _ ███○████○███x██○████○████
○ █████○████○███x██○████○████○████
██████████████████████████████████

```

```

4010 DIM B(12)
4020 DIM C(12)
4030 LET B$="4=*1K8,OBHER"
4040 FOR B=1 TO 12
4050 LET B(B)=CODE B$(B)
4060 LET C(B)=116-B(B)
4070 NEXT B
4080 LET C$="██████████████████████"
4090 LET B$="██████████████████████"
4100 PRINT AT 2,9;"A _ _ B _ _ C _ _ D _ _ E",,,
4110 FOR A=1 TO 5
4120 PRINT TAB 8;B$;TAB 6;A;TAB 24;A;TAB 8;C$
4130 NEXT A
4140 PRINT AT 20,9;"A _ _ B _ _ C _ _ D _ _ E"
4150 GOSUB 100
4160 LET B$="SF9/"
4170 PRINT AT 0,2;"DO_YOU_WANT_TO_GO_
FIRST?_Y/N"
4180 LET Z$=INKEY$
4190 IF Z$="" THEN GOTO 4170
4200 PRINT AT 0,2;"HIGH_JUMP_BY_GRAHAM_
CHARLTON"
4210 IF Z$="Y" THEN GOTO 3000
4220 GOTO 1000

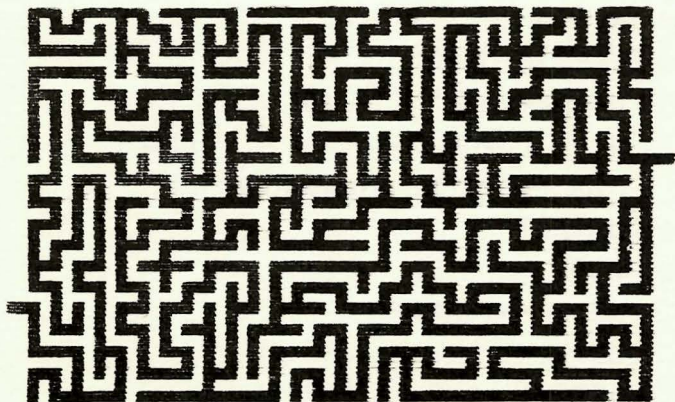
```

Maze Master

This fascinating program draws a maze, right before your eyes. There will be only one path through the maze, and you travel *on* the line, and not in the spaces as in most maze games.

You start your conquest of the maze on the left hand side, where the small piece is sticking out, and work your way to the right hand side.

The computer only generates the maze, it does not solve it. Use the printer to dump the maze onto paper, then solve it (if you can) with a pen. This program is good to watch and makes a great demonstration program.



Program 15: Maze Master

```

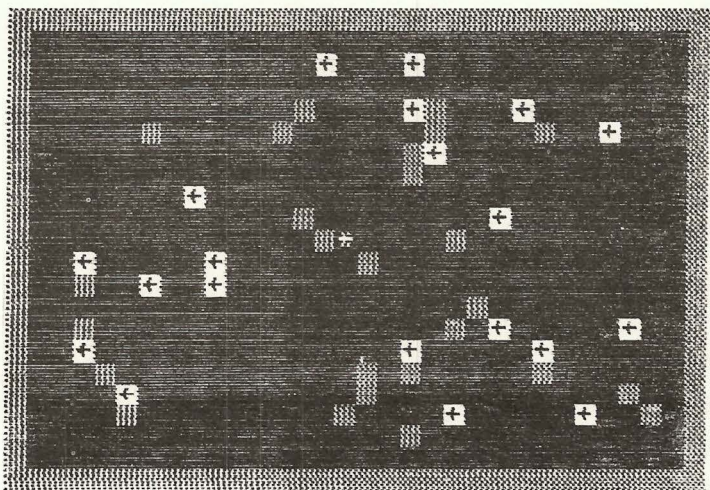
10  FAST
20  DIM A$(63,43)
30  FOR A=1 TO 63
40  LET A$(A,1)="2"
50  LET A$(A,2)="2"
60  LET A$(A,42)="2"
70  LET A$(A,43)="2"
80  NEXT A
90  FOR A=1 TO 43
100 LET A$(1,A)="2"
110 LET A$(2,A)="2"
120 LET A$(62,A)="2"
130 LET A$(63,A)="2"
140 NEXT A
150 SLOW
170 LET X=4
180 LET Y=4+2*INT (RND*18)
190 PLOT X-1,Y
200 PLOT X-2,Y
210 LET A$(X,Y)="2"
220 IF A$(X+2,Y)<>"_" AND A$(X-2,Y)
    <>"_" AND A$(X,Y+2)<>"_" AND
    A$(X,Y-2)<>"_" THEN GOTO 400
230 UNPLOT X,Y
240 PLOT X,Y
250 LET R=INT (RND*4)
260 LET C=X+2*(R=0)-2*(R=1)
270 LET D=Y+2*(R=2)-2*(R=3)
280 IF A$(C,D)<>"_" THEN GOTO 230

```

```
290 LET E=(C+X)/2
300 LET F=(D+Y)/2
310 LET A$(C,D)="1"
320 LET A$(E,F)="1"
340 PLOT E,F
350 LET X=C
360 LET Y=D
370 GOTO 220
400 LET A$(X,Y)="2"
410 PLOT X,Y
420 IF A$(X+1,Y)="1" THEN GOTO 500
430 IF A$(X-1,Y)="1" THEN GOTO 550
440 IF A$(X,Y+1)="1" THEN GOTO 600
450 IF A$(X,Y-1)="1" THEN GOTO 650
460 LET Y=4+2*INT (RND*18)
470 PLOT 61,Y
480 PLOT 62,Y
490 STOP
500 LET A$(X+1,Y)="2"
510 LET X=X+2
540 GOTO 220
550 LET A$(X-1,Y)="2"
560 LET X=X-2
590 GOTO 220
600 LET A$(X,Y+1)="2"
610 LET Y=Y+2
640 GOTO 220
650 LET A$(X,Y-1)="2"
660 LET Y=Y-2
690 GOTO 220
```

Zombies

You are trapped on an island inhabited by human-hating zombies (the plus signs). Your only hope of salvation is to lure them into the swamps (graphic A's), where they will drown most horribly. The zombies are pretty dumb, so they don't see the swamps until it is too late. However, they do see you, and head towards you whenever they can. When they get you, they'll pounce, and the game will be over. Wait until all the surviving plus signs have moved before you enter your move. You move by using the keys "5", "6", "7" and "8" and you move in the direction of the arrows on those keys.



Program 16: Zombies

```

10  GOTO 1000
20  LET Z$=INKEY$
30  IF Z$="" THEN GOTO 20
40  PRINT AT E,F;"■"
50  LET E=E+(Z$="6" AND E<20)-(Z$="7" AND
    E>1)
60  LET F=F+(Z$="8" AND F<30)-(Z$="5" AND
    F>1)
70  PRINT AT E,F;"* "
80  FOR A=1 TO 20
90  IF A(A)=0 THEN GOTO 190
100 PRINT AT A(A),B(A);"■"
110 IF ABS (E-A(A))>ABS (F-B(A)) THEN GOTO
    140
120 LET B(A)=B(A)-(B(A)>F)+(B(A)<F)
130 GOTO 150
140 LET A(A)=A(A)-(A(A)>E) + (A(A)<E)
150 PRINT AT A(A),B(A);
160 IF VAL A$=151 THEN GOTO 210
170 IF VAL A$=8 THEN LET A(A)=0
180 IF A(A) THEN PRINT "+ "
190 NEXT A
200 GOTO 20
210 FOR A=140 TO 191
220 PRINT AT E,F;CHR$ A
230 NEXT A
240 STOP

```

```

1000 PRINT "████████████████████████████████████████
██████████████████████████████████████████████████
██████"

1010 FOR A=1 TO 20
1020 PRINT "███ ████████████████████████████████████
███████████████████████████████████████████████"

1030 NEXT A
1040 PRINT "██████████████████████████████████████████
███████████████████████████████████████████████
██████"

1050 LET A$="PEEK (PEEK 16398 + 256*PEEK
16399)"

1060 DIM A(20)
1070 DIM B(20)
1080 LET E=11
1090 LET F=16
1100 FOR A=1 TO 30
1110 PRINT AT INT (RND*16)+2,
INT (RND*27)+2;"███"

1120 NEXT A
1130 FOR A=1 TO 20
1140 LET A(A)=11+INT (RND*9)*SGN(RND-.5)
1150 LET B(A)=16+INT (RND*14) *SGN (RND-.5)
1160 IF A(A)> 7 AND A(A)< 15 AND B(A)> 11 AND
B(A)< 20 THEN GOTO 1140
1170 PRINT AT A(A),B(A);"+ "
1180 NEXT A
1190 PRINT AT E,F;"* "
1200 GOTO 20

```


Poker Dice

Your Timex Sinclair computer always goes first in this version of Poker Dice. Lines 1000 to 1070 set up the variables and arrays, with 1000 reading "9TJQKA". Lines 20 to 40 are those which produce a random throw.

Lines 50 to 150 are for the computer's turn, with the subroutine from 500 to 580 deciding on whether it wishes to throw some of the dice again. You may wish to add a delay loop here to give the impression the computer is "thinking hard."

Lines 230 to 310 are for the player. If you wish to throw dice two, three and four again, enter "234". Enter "0" if you do not wish to change any of them.

You may like to add a routine to find out who has won. This is liable to be longer than the program which plays the game, and pretty slow besides.

After each game, the computer pauses for a short while, then begins a new game.

TS1000	PLAYER
S T T T K	K T K K K
T T T T S	K A K K Q
T T T T Q	K A K K A

Program 17: Poker Dice

```
10  GOTO 1000
20  LET A(A)=INT (RND*6)+1
30  LET B$(2*A)=A$(A(A))
40  RETURN
50  FOR A=1 TO 5
60  GOSUB 20
70  LET B(A(A))=B(A(A))+1
80  NEXT A
90  PRINT TAB 10;B$
100 PRINT
110 FOR G=1 TO 2
120 GOSUB 500
130 PRINT TAB 10;B$
140 PRINT
150 NEXT G
160 PRINT TAB 12;"PLAYER"
170 PRINT
180 FOR C=1 TO 5
190 LET C$(2*C)=A$ (INT (RND*6)+1)
200 NEXT C
210 PRINT TAB 10;C$
220 PRINT
230 FOR G=1 TO 2
240 INPUT D$
250 IF D$="" THEN GOTO 300
260 FOR C=1 TO LEN D$
270 LET C$(2*VAL D$(C))=A$(INT (RND*6)+1)
280 NEXT C
```

```
290 PRINT TAB 10;C$
300 PRINT
310 NEXT G
320 FOR A=1 TO 200
330 NEXT A
340 CLS
350 RUN
500 DIM R(6)
510 FOR B=1 TO 6
520 IF B(B)<2 THEN LET R(B)=1
530 NEXT B
535 DIM B(6)
540 FOR A=1 TO 5
550 IF R(A(A))=1 THEN GOSUB 20
560 LET B(A(A))=B(A(A))+1
570 NEXT A
580 RETURN
1000 LET A$="9TJQKA"
1010 DIM B$(10)
1020 DIM A(5)
1030 DIM B(6)
1040 DIM C$(10)
1050 PRINT TAB 13;"TS1000"
1060 PRINT
1070 GOTO 50
```

Antimind

Instead of setting up a code for you to guess, as in most computer games of MASTERMIND, this program sets the computer to solving the code you have given it.

Your four-digit code must be made up from the numbers one to six. Once the computer has made a guess, enter its score as a single string (such as "21" or "03"), the first digit being the number of 'blacks' (correct digit in correct place), the second being the 'whites' (correct digit, but not in the right position).

The program is rather slow. It can take up to four minutes to make a decision (although this is most unlikely), although most moves will take about a minute. At times, it can decide in seconds. The computer can take up to nine guesses to crack your code. Line 350 is just there to get your hopes up. Always run the program in **FAST**.

The main lines are 240 and 260, which allow the computer to cope with codes which have repeated digits, and lines 370 to 400 which speed the program up by jumping out of loops. If you cheat, line 450 will tell you so. The loops Z,Y,W and X correspond to each digit in the code. You may change the program so you can have a code with the digits from one to nine in it. Just change lines 30, 70, 80, 90 and 100. This, however, will make the game very slow.

```

          BLACK WHITE
0440000H 000000H 000000H
4440000H 444000H 444000H
4440000H 444000H 444000H
4440000H 444000H 444000H

```

I GUESSED YOUR CODE IN 7 TRIES

Program 18: Antimind

```

10 PRINT TAB 8;"BLACK_WHITE"
20 DIM A(10,5)
30 LET Z$="123456123456" (INT (RND*6)+1 TO )
40 LET S=1
50 LET G=1
60 LET Q=1
70 FOR Z=S TO 6
80 FOR Y=S TO 6
90 FOR X=S TO 6
100 FOR W=S TO 6
110 LET A(G,1)=Z
120 LET A(G,2)=Y
130 LET A(G,3)=X
140 LET A(G,4)=W
150 IF Q=1 THEN GOTO 290
160 FOR T=1 TO G-1
170 LET E=4
180 LET K=A(T,1)
190 LET L=A(T,2)
200 LET M=A(T,3)
210 LET N=A(T,4)

```

```

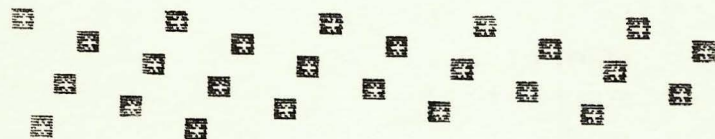
220 FOR P=1 TO 4
230 LET R=A(G,P)
240 LET E=E-(R<>K AND R<>L AND R<>M
    AND R<>N)
250 NEXT P
260 LET E=E+9*((Z=K)+(Y=L)+(X=M)+(W=N))
270 IF E<>A(T,5) THEN GOTO 410
280 NEXT T
290 PRINT AT G,0;Z$(Z);" _";Z$(Y);" _";Z$(X);
    "_";Z$(W);
300 INPUT A$
310 PRINT "_ _";A$(1);" _ _ _ _";A$(2)
320 LET A(G,5)=VAL A$
330 LET Q=0
340 LET G=G+1
350 IF G=11 THEN PRINT "I GIVE UP";D
360 IF VAL A$=40 THEN PRINT ", "I_GUESSED_
    YOUR_CODE_IN_"G-1;"_TRIES";D
370 IF VAL A$=0 THEN LET S=S+1
380 IF VAL A$<10 THEN GOTO 440
390 IF VAL A$<20 THEN GOTO 430
400 IF VAL A$<30 THEN GOTO 420
410 NEXT W
420 NEXT X
430 NEXT Y
440 NEXT Z
450 PRINT ", "I_THINK_YOU_CHEATED";D

```

Nim

In this computer adaptation of an allegedly ancient oriental game, you have to take a number of objects away from those left in front of you, trying to be the player who removes the last object in order to win. There is a maximum number you can take on any move, and in this program the maximum lies between three and five. The computer sets the maximum for each game.

The algorithm used is very simple. You'll see in line 180 that if the computer sees it is in a losing position, it sends control to line 240 where it subtracts a random number of pieces. You may wish to add a couple of lines to make it deliberately make mistakes to give you more chance of winning. On your turn, just press the key relating to the number of pieces you want to remove.



```
THERE ARE 25 LEFT, THE MAXIMUM
YOU CAN TAKE IS 5
HOW MANY WILL YOU TAKE?
```

Program 19: Nim

```

10 LET A=INT (RND*12)+20
20 LET B=INT (RND*3)+3
30 LET D=B+1
40 CLS
50 IF A=0 THEN PRINT AT 10,13;"I WON";Q
60 GOSUB 260
70 PRINT AT 10,0;"THERE ARE ";A;" LEFT,
  THE MAXIMUM"
80 PRINT AT 12,6;"YOU CAN TAKE IS ";B
90 PRINT AT 14,4;"HOW MANY WILL YOU
  TAKE?"
100 IF INKEY$="" THEN GOTO 100
110 LET C=VAL INKEY$
120 LET A=A-C
130 LET N=A
140 CLS
150 IF A=0 THEN PRINT AT 10,12;"YOU WON";Q
160 GOSUB 260
170 LET C=0
180 IF INT (A/D)*D=A THEN GOTO 240
190 LET A=A-(A-INT (A/D)*D)
200 PRINT AT 10,12;"I TOOK ";N-A
210 FOR E=1 TO 50
220 NEXT E
230 GOTO 40
240 LET A=A-INT (RND*B)-1
250 GOTO 200

```



```
260 FOR E=1 TO A
270 PRINT "*" _ _ _ _ _ ";
280 NEXT E
290 RETURN
```

Hundred Up

In this game, you and your computer take turns to throw a die as many times as you like, each time adding the result of the throw to your score.

However, if you lose all the points you have scored on that turn. You may stop throwing at any time, keeping all the points to date, and then it is your opponent's turn. The first to score 100 points wins.

The computer's thinking is containing in line 320. Here, the computer will throw if it is more than 30 behind, its score is less than 13 for this go, it has had less than two throws, the player has more than 86. It will stop if it has more than 100. Try different strategies in this line, to see the effect this has on the computer's game. To play, just press "Y" or "N" to the question asked by line 90.

PLAYER

11
11
23
33
46
59

TS1000

0
0
15
29
44
58

Program 20: Hundred Up

10 LET A=0

20 LET B=0

```

30 LET E=0
40 PRINT AT 0,7;"PLAYER";TAB 20;"TS1000"
50 LET C=0
60 LET D=0
70 LET F=0
80 LET E=E+1
90 PRINT AT 21,4;"PRESS_Y_TO_ROLL_N_TO_
STOP"
100 LET Z$=INKEY$
110 IF Z$="" THEN GOTO 100
120 PRINT AT 21,4;" _ _ _ _ _ _ _ _ _ _
_ _ _ _ _ _ _ _ _ _"
130 IF Z$="N" THEN GOTO 210
140 PRINT AT E,9;" _ _"
150 LET G=INT (RND*6)+1
160 IF G=1 THEN GOTO 200
170 LET D=D+G
180 PRINT AT E,9;D
190 GOTO 90
200 LET D=0
210 LET B=B+D
220 PRINT AT E,9;B
230 IF B >= 100 THEN PRINT AT E+1,7;
"YOU WON";Q
240 PRINT AT E,23;" _ _"
250 LET H=INT (RND*6)+1
260 LET F=F+1
270 IF H=1 THEN GOTO 340
280 LET C=C+H
290 PRINT AT E,23;C

```

```
300 FOR J=1 TO 20
310 NEXT J
320 IF (B-A-C<30 AND C>13 AND F>2 AND
    B<86) OR A+C>=100 THEN GOTO 350
330 GOTO 240
340 LET C=0
350 LET A=A+C
360 PRINT AT E,23;A
370 IF A>=100 THEN PRINT AT E+1,22;"I WON";Q
380 GOTO 50
```

Tic-Tac-Toe

This is an elaborate, and hard to beat, Tic-Tac-Toe program. You have the option of going first or second. You are the O's, the computer is the X's. Wait for the board to change before entering your move, which you do simply by entering the number (one to nine) of the square into which you want to move.

Note how this program makes use of the TS1000's "string-slicing" ability (for example, lines 20 to 40). Play this game in **SLOW** mode. The program does not say who has won the game, so each game has to be played out completely.

Program 21: Tic-Tac-Toe

```

10  GOTO 1000
20  PRINT AT 3,11;A$(1 TO 11)
30  PRINT AT 7,11;A$(13 TO 23)
40  PRINT AT 11,11;A$(25 TO 34)
50  RETURN
60  GOSUB 20
70  IF D=9 THEN GOTO 1000
80  IF D=3 AND (A(1)=1 AND A(9)=1 OR A(3)=1
    AND A(7)=1) THEN LET B$=C$

```

```

90  LET D=D+1
95  FOR E=-2 TO 2 STEP 4
100 FOR A=1 TO 3
110 IF A(A*3-2)+A(A*3-1)+A(A*3)=E THEN GOTO
    190
120 IF A(A)+A(A+3)+A(A+6)=E THEN GOTO 210
130 NEXT A
140 IF A(1)+A(5)+A(9)=E THEN GOTO 230
150 IF A(3)+A(5)+A(7)=E THEN GOTO 250
155 NEXT E
160 FOR C=1 TO 9
165 LET B=VAL B$(C)
170 IF A(B)=0 THEN GOTO 260
180 NEXT C
190 LET B=(A*3-2)*(NOT A(A*3-2))+(A*3-1)*
    (NOT A(A*3-1))+(A*3)*(NOT A(A*3))
200 GOTO 260
210 LET B=A*(NOT A(A))+A+3*(NOT
    A(A+3))+A+6*(NOT A(A+6))
220 GOTO 260
230 LET B=(NOT A(1))+5*(NOT A(5))+9*(NOT A(9))
240 GOTO 260
250 LET B=3*(NOT A(3))+5*(NOT A(5))+7*(NOT A(7))
260 LET A$(B*4-2)="X"
270 LET A(B)=-1
280 GOSUB 20
290 IF D=9 THEN GOTO 1000
300 LET Z$=INKEY$
310 IF Z$="" THEN GOTO 300

```

```

320 LET A$(VAL Z$*4-2)="O"
330 LET A(VAL Z$)=1
340 LET D=D+1
350 GOTO 60
1000 LET A$="_1_█_2_█_3_█_4_█_5_█
      _6_█_7_█_8_█_9"
1010 LET B$="573914826"
1020 IF RND<.2 THEN LET B$="159378246"
1030 LET C$="431978625"
1040 DIM A(9)
1050 PRINT AT 16,2;"DO_YOU_WANT_TO_GO_
      FIRST?_(Y/N)"
1060 LET Y=INKEY$
1070 IF Y$="" THEN GOTO 1060
1080 PRINT AT 16,2;"_ _ _ _ _ _ _ _ _ _
      _ _ _ _ _ _ _ _ _ _"
1090 FOR A=1 TO 6
1100 PRINT AT A*2,11;"_ _ _ █ _ _ _ █"
1110 NEXT A
1120 PRINT AT 5,11;"█ █ █ █ █ █ █ █ █ █"
1130 PRINT AT 9,11;"█ █ █ █ █ █ █ █ █ █"
1140 LET D=0
1150 IF Y$="Y" THEN GOTO 280
1160 GOTO 60

```

Alternate Version

Here is another version of Tic-Tac-Toe showing a completely different programming approach. A comparison of the two programs is very interesting.

Unlike many programs of this game, it is possible to beat this version occasionally, although a draw or a loss is the most probable outcome. Some moves will be made more or less instantly, while others will take ten seconds or so. The strategy is to look for a winning move, or if no winning move is possible, look for a possible winning move by the player and take action to block it if found. If such a move is not found, a move is chosen at random. The program will know when a game has been won or drawn.

All responses are detected using **INKEY\$**, so you do not need to press **ENTER**. All responses should consist of one character. You will be asked, at the beginning, if you want to go first. If you do, press "Y", if not "N".

Here is the numbering system for the board:

1	2	3
4	5	6
7	8	9

A discussion of the program follows the listing.

Program 21A: Tic-Tac-Toe

```

2  RAND
5  LET X=0
10 DIM B$(9)
12 PRINT AT 0,0;"DO_YOU_WANT_FIRST_GO_
    (Y_OR_N)?"
14 LET A$=INKEY$
16 IF A$ < > "Y" AND A$ < > "N" THEN GOTO 14
18 CLS
20 PRINT AT 5,13;"1"█2█3";TAB 13;"█ █
    █ █ █";TAB 13;"4"█5█6";TAB 13;"
    █ █ █ █ █";TAB 13;"7"█8█9"

```



```

25  IF A$="N" THEN GOTO 100
30  PRINT AT 12,12;"YOUR_GO"
40  LET A$=INKEY$
50  IF A$<"1" OR A$>"9" THEN GOTO 40
60  IF B$(CODE A$-28)>"_" THEN GOTO 40
70  LET B$(CODE A$-28)="O"
80  PRINT AT CODE "■ ■ ■ ■ ■ ■ ■ ■ ■ ■" (CODE
    A$-28), CODE "$?)$?)$?)" (CODE A$-28);B$
    (CODE A$-28)
85  LET A$="OOO"
90  GOTO 300
100 REM **TS1000 MOVE"
110 PRINT AT 12,12;"_MY_GO_"
111 IF B$(5)="_" THEN LET F=5
112 IF B$(5)="_" THEN GOTO 240
114 FOR B=1 TO 2
116 LET C$="XO"(B)+"XO"(B)
120 FOR A=1 TO 70 STEP 3
130 LET A$="15919559135757337512313223145656
    446578979898714747117425828558236939669
    3"(A TO A+2)
140 LET D=CODE A$-28
150 LET E=CODE A$(2)-28
160 LET F=CODE A$(3)-28
170 IF B$(D)+B$(E)=C$ AND B$(F)="_" THEN
    GOTO 240
180 NEXT A
190 NEXT B
193 LET A$=""

```


Programming Notes

Line 2: Sets **RAND** so that a fresh series of random numbers is generated.

Line 5: Initializes the value of **X**, the variable which counts the numbers of moves made. This is useful for determining when a draw occurs.

Line 10: Initializes **B\$** to 9 spaces. **B\$** is a string array which contains the state of the board. The element subscripts correspond to the layout of the board on screen, e.g. **B\$(5)** corresponds to the central square of the board on the screen.

Line 12: Asks the player who goes first.

Line 14: Scans keyboard, stores in **A\$**.

Line 16: Continues scanning keyboard until a **Y** (for **YES**) or an **N** (for **NO**) is detected, then carries on accordingly.

Line 18: Clears the question from the screen.

Line 20: Prints the initial state of the board with the numbers to tell the user which keys to press for each position on the board.

Line 25: If the player declined first go (player pressed **N**) jumps to line 100 for the computer to move.

Line 30: Prints message saying it's the player's turn.

Line 40: Scans keyboard to find player's move and stores in **A\$**.

Line 50: If **A\$** is not a valid move, i.e. the key pressed is not in the range 1 to 9 then continues scanning until a suitable character is returned.

Line 60: If the square on which the player wants to put an **O** already has an **O** or an **X** then re-enters.

Line 70: Marks an **O** in the square chosen by the player (makes the appropriate element of **B\$** an **O**).

Line 80: Prints the move. The **X** and **Y** co-ordinates for the **PRINT AT** statement are obtained by extracting the **CODE** of the elements of the string literals in the **PRINT AT** statement.

Note how (**CODE A\$-28**) is used to decode the strings into numbers rather than using **VAL A\$**: the reason is that **CODE A\$-28** is about 50% faster than **VAL**, and while it

doesn't make much difference in this particular example, when we get into a long loop later on we'll begin to see the difference.

Line 85: Sets A\$ to three O's for the win-checking routine. A\$ will also act as a flag to tell the routine where to come back to if the game does not end.

Line 90: Sends the program to a routine at line 300 to see if the game has been won or drawn. Why not **GOSUB**? We may not return if the game is won or drawn, so the **GOSUB** address will be left on the stack (all right, the **RETURN** address) and they will pile up after a few games. Since there is not much memory left out of 2K, if you are using a 2K machine you will soon run out of free memory and will have to go through the long-winded process of ending **RETURN** manually to clear up the **GOSUB** stack. So we use the **GOTO** facility instead.

Line 100: A **REM** statement to identify the computer-move part of the program. If you are going to make any changes to the program that will require extra memory space, this line will be the first target for deletion.

Line 110: Displays the fact that it's the computer's turn.

Lines 111 & 112: If the center square is available, that is the computer's move. F is a flag saying which move the computer has chosen to make. It is used by the computer-move-printing routine later on.

Lines 114 & 190: Consist of the routine which will either block an opponent's winning chance or connect up a winning line for the computer. It looks at the board and if it finds two identical characters in a line the time for action has arrived. To sort out whether it is a possible winning move for the player or the computer the loop is executed twice, first looking for two X's (computer) and second looking for two O's. To prevent the two identical characters spotted being two spaces, C\$ indicates the character to be searched for. A\$ is a string of three characters extracted from the string constant in line 130. The first two characters specify which two squares to check, and if the result is positive the third character of A\$ says where to put the computer's X.

Lines 140, 150 & 160: Convert the characters from A\$ into numbers. Note how **CODE** A\$-28 is used rather than **VAL** A\$ because of speed requirements. This can only be used with numbers of course. Numeric statements cannot be decoded in this way nor can variables.

Lines 193 to 240: If a winning move or blocking an opponent's move is not possible, a move is made at random by gathering together all the squares without an X or an O and choosing one of them at random.

Line 250: Prints the move.

Line 260: As line 85, but sets A\$ for three X's.

Line 300: Checks for a winning situation and if found, goes to the game-end routines.

Line 304: Increments move counter. If this reaches 9 no further moves are possible since there are only nine squares in the board, and the game is a draw unless there is a winning combination on the board.

Line 310: If the game has not come to an end, returns for the player's/computer's move, which one depends on whether A\$ is "OOO" or "XXX".

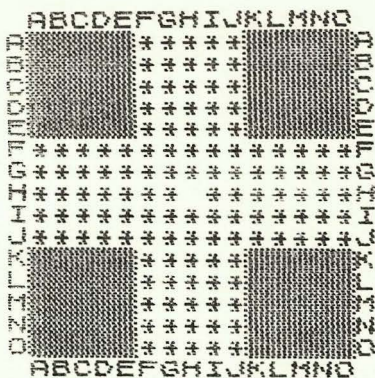
Lines 320 to 350: Print who won, or if neither, announce a draw.

Solitaire

The object of this game is to be left with only one peg on the board, preferably with that peg in the center hole. A peg is removed by jumping over it with another peg. To move a peg, enter a three-character string, with the following direction after the coordinates of the peg:

- 5-To move peg left
- 6-To move peg down
- 7-To move peg up
- 8-To move peg right

You'll see that the arrows on these keys point in the required directions.



SOLITAIRE

ONE LEFT


```

PEG_LEFT_____6_
TO_MOVE_PEG_DOWN_____
____7_TO_MOVE_PEG_UP_____
_____8_TO_MOVE_PEG_
RIGHT"
60 PRINT AT 17,1;"ALSO,_ENTER"; AT 19,1;
   """"S""_TO_RETURN_TO_COMMAND_
   MODE","_""R""_TO_RE-RUN_PROGRAM"
90 PAUSE 40000
95 DIM M$(3)
100 CLS
105 PRINT AT 5,20;"SOLITAIRE";AT 6,20;"■■■■
   ■■■■";AT 9,20;"PEGS_LEFT"
110 LET P=256*PEEK 16397+PEEK
   16396
115 LET N=124
125 FOR I=1 TO 5
130 FOR J=1 TO 5
135 PRINT AT I,J;"■";AT I+10,J;"■";AT I,J+10;
   "■";AT I+10,J+10;"■"
140 NEXT J
145 NEXT I
150 FOR I=1 TO 15
155 FOR J=6 TO 10
160 PRINT AT I,J;"*";AT J,I;"*"
165 NEXT J
170 PRINT AT 0,I;CHR$(37+I);AT I,0;CHR$(37+I);AT
   16,I;CHR$(37+I);AT I,16;CHR$(37+I);AT 8,8;"_"
175 NEXT I

```



```

180 INPUT M$
185 IF M$(1)="R" THEN GOTO 100
190 IF M$(1)="S" THEN STOP
195 LET S=P+33*CODE M$(1)+CODE M$(2)-1257
200 LET A=CODE M$(3)
205 LET F=S+(66 AND A=34)+(2 AND A=36)-(66
AND A=35)-(2 AND A=33)
210 IF PEEK S<>23 OR PEEK ((S+F)/2)<>23 OR
PEEK F<>0 THEN GOTO 180
215 POKE S,0
220 POKE F,23
225 POKE (S+F)/2,0
230 LET N=N-1
235 PRINT AT 10,20;">>_";N;"_<<_"
240 IF N>=2 THEN GOTO 180
245 PRINT AT 21,0;"PRESS_ENTER_TO_PLAY_
AGAIN"
250 INPUT M$
255 IF M$="" THEN GOTO 100
260 STOP

```

Sample Run

SOLITAIRE

THE OBJECT OF THE GAME IS TO BE LEFT WITH ONLY ONE PEG AND PREFERABLY WITH IT IN THE CENTER OF THE BOARD. A PEG IS REMOVED BY JUMPING OVER IT WITH ANOTHER PEG. TO MOVE A PEG,

118 / Chapter 22

ENTER (AS A 3 CHARACTER STRING)
ITS CO-ORDINATES FOLLOWED BY:-

- 5 TO MOVE PEG LEFT
- 6 TO MOVE PEG DOWN
- 7 TO MOVE PEG UP
- 8 TO MOVE PEG RIGHT

ALSO, ENTER

"S" TO RETURN TO COMMAND MODE

"R" TO RE-RUN PROGRAM

2

Part

UTILITY AND APPLICATIONS PROGRAMS

Address Book

The days of pencil and paper are past. At least, that's the impression given by this program, which enables you to save your friends' names and addresses within the program where they can be either displayed, listed, printed out, or saved on tape for future reference. The program is menu-driven.

Program 23: Address Book

```

5  REM "ADDRESS_BOOK"
15  SLOW
20  DIM N$(15)
25  DIM A$(51,95)
30  LET N=1
40  CLS
45  PRINT AT 5,0;"PRESS:"",,,,,,"L"" _TO_LIST_
    ALL_NAMES_IN_BANK","A"" _TO_
    DISPLAY_ADDRESSES","S"" _TO_SAVE_
    ON_TAPE","C"" _TO_RETURN_TO_
    COMMAND_MODE","E"" _TO_ENTER_
    NAME_TO_BANK","D"" _TO_DELETE_
    NAME_FROM_BANK","P"" _TO_PRINT_
    ADDRESS"

```

```

50  IF INKEY$="E" THEN GOSUB 100
55  IF INKEY$="D" THEN GOSUB 300
60  IF INKEY$="P" THEN GOSUB 500
65  IF INKEY$="A" THEN GOSUB 700
70  IF INKEY$="L" THEN GOSUB 900
75  IF INKEY$="C" THEN STOP
80  IF INKEY$="S" THEN SAVE "ADDRESS.S"
85  GOTO 45
100 CLS
105 PRINT AT 5,0;"ENTER:"",,"NAME_ _ _ _ _";
110 INPUT A$(N, TO 15)
115 PRINT A$(N, TO 15), "ADDRESS_ _ _";
120 INPUT A$(N,16 TO 35)
125 PRINT A$(N,16 TO 35), "TOWN_ _ _ _ _";
130 INPUT A$(N,36 TO 55)
135 PRINT A$(N,36 TO 55), "COUNTY_ _ _ _";
140 INPUT A$(N,56 TO 67)
145 PRINT A$(N,56 TO 67), "POST_CODE_";
150 INPUT A$(N,68 TO 75)
155 PRINT A$(N,68 TO 75), "PHONE_ _ _ _ _";
160 INPUT A$(N,76 TO )
165 PRINT A$(N,76 TO )
170 LET N=N+1
175 IF INKEY$="" THEN GOTO 175
180 CLS
185 RETURN
300 CLS
305 PRINT AT 5,0;"NAME_TO_BE_DELETED?"
310 INPUT N$

```

```
315  FOR I=1 TO 50
320  IF N$ <> A$(I, TO 15) THEN GOTO 345
325  FOR J=1 TO 50
330  LET A$(J)=A$(J+1)
335  NEXT J
337  LET N=N-1
340  GOTO 350
345  NEXT I
350  CLS
355  RETURN
500  CLS
505  PRINT AT 5,0;"NAME ENQUIRY?"
510  INPUT N$
515  FOR I=1 TO 50
520  IF A$(I, TO 15) <> N$ THEN GOTO 535
525  PRINT ,,A$(I, TO 15),,A$(I,16 TO 35),A$(I,36 TO
      55),A$(I,56 TO 67),,A$(I,68 TO 75),,A$(I,76 TO 95)
527  LPRINT A$(I, TO 15),,A$(I,16 TO 35),A$(I,36 TO
      55), A$(I,56 TO 67),,A$(I,68 TO 75),,A$(I,76 TO 95)
530  GOTO 540
535  NEXT I
540  IF INKEY$="" THEN GOTO 530
545  CLS
550  RETURN
700  FOR I=1 TO N-1
705  CLS
710  PRINT AT 7,0;A$(I, TO 15),,A$(I,16 TO 35),A$(I,36
      TO 55),A$(I,56 TO 67),,A$(I,68 TO 75),,A$(I,76 TO
      95)
```

```
715  IF INKEY$="" THEN GOTO 715
720  NEXT I
725  CLS
730  RETURN
900  CLS
905  FOR I=1 TO N
910  PRINT AT 20,0;A$(I, TO 15)
915  SCROLL
920  NEXT I
925  CLS
930  RETURN
```



```
50 INPUT Q$
55 IF Q$="S" THEN STOP
60 IF Q$="C" THEN GOSUB 200
65 IF Q$="E" THEN GOSUB 300
70 IF Q$="V" THEN GOSUB 500
75 IF Q$="R" THEN GOSUB 600
80 IF Q$="L" THEN GOSUB 100
85 GOTO 40
100 CLS
105 PRINT AT 0,5;S$;AT 2,5;"THE _VALUE_ OF _
RESISTORS _ARE _MARKED _BY _FOUR _
COLORED _BANDS _ _ _WHICH _INDICATE _
THEIR _RESISTANCE _VALUE _AND _THEIR _
TOLERANCE."
110 PRINT AT 6,0;R$
115 PRINT AT 12,0;" _ _ _ _ _THE _FIRST _BAND _
INDICATES _THE _FIRST _FIGURE _OF _THE _
VALUE, _THE _SECOND _BAND _INDICATES _
THE _SECOND _FIGURE _OF _THE _VALUE, _
AND _THE _THIRD _BAND _INDICATES _HOW _
MANY _ZEROS _THE _VALUE _CONTAINS."
120 PRINT AT 19,5;"THE _FOURTH _BAND _
INDICATES _ _THE _TOLERANCE, _I.E. _HOW _
ACCURATEIT _IS."
125 INPUT Q$
130 GOSUB 200
135 IF Q$="M" THEN GOTO 150
140 GOSUB 300
150 GOTO 40
```

```

200  CLS
205  PRINT S$;AT 1,18;"COLOR_VALUE"
210  FOR I=1 TO 10
215  PRINT AT 1+2*I,18;C$(I);TAB 27;I-1
220  NEXT I
225  PRINT AT 5,0;" _ _ _ THE _ FIGURE" ,,"THAT _
    EACH _ OF" ,,"THE _ FIRST _ THREE" ,,"
    "COLORED _ BANDS" ,,"REPRESENTS _
    IS" ,,"GIVEN _ BY _ THIS" ,,"TABLE."
230  PRINT AT 15,3;"THE _ UNIT _ OF" ,,"
    "RESISTANCE _ IS" ,,"OHMS."
235  INPUT Q$
240  IF Q$="M" THEN GOTO 270
250  CLS
255  PRINT S$;AT 4,4;"THE _ TOLERANCE _ OF _ THE _
    _ _ _ _ _ RESISTOR _ IS _ GIVEN _ BY _
    THE _ _ _ _ _ FOLLOWING _ COLOR _
    VALUES _ _ _ _ _ OF _ THE _
    FOURTH _ BAND:"
260  PRINT AT 10,0;"COLOR _ _ TOL.
    (PERCENTAGE)" ,,"BROWN _ _ _ _ _ 1" ,,"
    "RED _ _ _ _ _ 2" ,,"GOLD _ _ _ _ _
    5" ,,"SILVER _ _ _ 10"
265  PRINT AT 17,3;"THE _ TOLERANCE _ BAND _ IS _
    _ _ _ _ _ OFTEN _ ABSENT _ FROM _ THE _
    _ _ _ _ _ RESISTOR. _ _ IN _ THIS _
    CASE _ _ _ _ _ ASSUME _ A _ 20 _
    PERCENT _ _ _ _ _
    TOLERANCE."

```

```

270 INPUT Q$
275 RETURN
300 CLS
305 PRINT AT 0,5;S$;AT
    3,0;"EXAMPLES."", "-----"
310 PRINT AT 5,0;R$
315 PRINT AT 13,0;"BAND_1__BAND__2__
    BAND_3__BAND_4"
320 INPUT Q$
325 PRINT AT 15,0;"GREEN__BLACK__
    BLACK__GOLD"
330 PRINT AT 17,2;"50_X_1_=_50_
    OHMS.";AT 18,2;"5_PERCENT_
    TOLERANCE.";AT 20,0;"PRESS__
    ENTER_", "OR_ENTER_"M" _FOR_
    MENU."
335 INPUT Q$
340 IF Q$="M" THEN GOTO 420
345 PRINT AT 15,0;"BLUE__GREEN__
    YELLOW__RED__"
350 PRINT AT 17,2;"65_X_10000_=_650,000_
    OHMS.";AT 18,2;"2_PERCENT_TOLERANCE."
355 INPUT Q$
360 IF Q$="M" THEN GOTO 420
365 PRINT AT 15,0;"GREY__VIOLET__
    GREEN__SILVER"
370 PRINT AT 17,2;"87_X_100000_=_8,700,000_
    OHMS._";AT 18,2;"10_PERCENT_
    TOLERANCE._"

```

```

375  INPUT Q$
380  IF Q$="M" THEN GOTO 420
385  PRINT AT 15,0;"BLACK_ _ _BROWN_ _ _
      BLACK_ _ _BROWN_"
390  PRINT AT 17,2;"01_X_1_ = _1_OHM. _ _ _ _
      _ _ _ _ _";AT 18,2;"1_PERCENT_
      TOLERANCE. _ _ _ _"

395  INPUT Q$
400  IF Q$="M" THEN GOTO 420
405  PRINT AT 15,0;"YELLOW_ _ _ORANGE_ _ _RED_
      _ _ _ _SILVER_"
410  PRINT AT 17,2;"43_X_100_ = _4300_OHMS. _
      _";AT 18,2;"10_PERCENT_TOLERANCE."

415  INPUT Q$
420  RETURN
500  CLS
505  PRINT AT 0,5;S$;AT 3,0;"TO_CONVERT_A_
      VALUE_TO_RESISTOR_ _COLOR_CODE."
510  PRINT AT 6,0;"ENTER_VALUE_OF_RESISTOR_
      (OHMS)"
515  INPUT V$
520  IF LEN V$ < 1 OR LEN V$ > 11 THEN GOTO 515
525  PRINT AT 8,0;R$;AT 15,4;V$;"_
      OHMS.";AT 17,0;"_ _ _ _BAND_1_ _ _BAND_
      2_ _BAND_3"
530  IF LEN V$ = 1 THEN LET V$ = "0" + V$
535  PRINT " _ _ _ _";C$(VAL V$(1)+1);" _ _
      ";C$(VAL V$(2)+1);" _ _";C$(LEN V$-1)
540  PRINT AT 21,4;"ENTER_ ""M"" _FOR_MENU."

```

```

545  INPUT Q$
550  IF Q$ <> "M" THEN GOTO 500
555  RETURN
600  CLS
605  PRINT AT 0,5;S$;AT 3,0;"TO_CONVERT_A_
      RESISTOR_COLOR_ _ _ _ _CODE_TO_A_
      VALUE.";AT 5,0;R$
610  DIM D$(3,6)
615  FOR I=1 TO 3
620  PRINT AT 10+2*I,0;"ENTER_COLOR_OF_
      BAND_";I;"_";
625  INPUT D$(I)
630  PRINT D$(I)
635  NEXT I
640  DIM N(3)
645  FOR I=1 TO 3
650  FOR J=1 TO 10
655  IF D$(I) < > C$(J) THEN GOTO 670
660  LET N(I)=J-1
665  GOTO 680
670  NEXT J
675  GOTO 690
680  NEXT I
685  PRINT AT 18,4;"VALUE_";N(1);N(2);
      "0000000000"(1 TO N(3));"_OHMS."
690  PRINT AT 21,4;"ENTER_""M""_FOR_MENU."
695  INPUT Q$
700  IF Q$ <> "M" THEN GOTO 600
705  RETURN

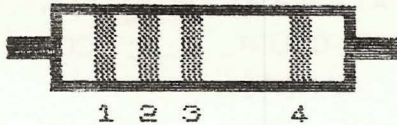
```

Sample Run

RESISTOR CODE.



EXAMPLES.



BAND 1 BAND 2 BAND 3 BAND 4

BLACK BROWN BLACK BROWN

01 X 1 = 1 OHM.
1 PERCENT TOLERANCE.

PRESS ENTER
OR ENTER "M" FOR MENU


```

ENTERED_AND_THEN__PRINTS_OUT__
THE_PUPILS_NAMES_IN__THE_ORDER__
OF_THEIR_RESULTS,_WITHTHEIR_RESULT__
WRITTEN_AS_A____
PERCENTAGE_AND_A_GRADE."
30 PRINT AT 9,5;"ENTER_FIRST_THE_NAME__
OF_THEPUPIL_AND_THEN_THEIR__
RESULT_IN__THE_EXAM.__AFTER_THE__
LAST_RESULTENTER_""_STOP_""_(SHIFT__
A).__IF_ANERROR_IS_MADE,_ENTER__
""N""_WHEN__""CORRECT?""_
APPEARS."
35 PRINT AT 15,5;"RESULTS_MAY_BE__
""SAVED""_ON__TAPE_AND_WHEN__
""RE-LOADED""_CAN_BE_OBTAINED_BY__
CALLING_FOR_THE_MENU_IMMEDIATELY."
40 PRINT AT 19,0;"ENTER:";AT 20,0;""""" _
TO_INPUT_NAMES_AND__
RESULTS","""M""_FOR_MENU"
45 INPUT Q$
50 IF CODE Q$=50 THEN GOTO 260
55 CLS
60 PRINT AT 10,0;"WHICH_GRADING__
SYSTEM";AT 12,0;"ENTER:"",""A""_FOR__
A,B,C",""B""_FOR__A,B,C,D,E"
65 INPUT Q$
67 LET G=9+(6 AND Q$="B")
70 CLS

```

```
75 PRINT AT 10,0;"ENTER_MAXIMUM_RESULT_  
IN_EXAM"  
80 INPUT M$  
85 PRINT AT 12,10;M$;AT 14,8;"CORRECT?"  
90 INPUT Q$  
95 IF CODE Q$=51 THEN GOTO 70  
100 LET M=INT ABS VAL M$  
105 FOR I=1 TO 150  
110 CLS  
115 PRINT AT 9,0;"ENTER_NAME_OF_PUPIL_";I  
120 INPUT N$(I)  
125 IF CODE N$(I)=227 THEN GOTO 170  
130 PRINT AT 11,0;N$(I);AT 14,0;"ENTER_PUPILS_  
EXAM_RESULT";  
135 INPUT R$  
140 PRINT AT 16,0;R$;AT 19,0;"CORRECT?"  
145 INPUT Q$  
150 IF CODE Q$=51 THEN GOTO 110  
155 LET M(I)=INT (VAL R$*100/M)  
160 LET N=I  
165 NEXT I  
170 FAST  
175 FOR J=2 TO N  
180 LET T$=N$(J)  
185 LET T=M(J)  
190 FOR I=J-1 TO 1 STEP -1  
195 IF M(I)>T THEN GOTO 215  
200 LET N$(I+1)=N$(I)
```

```

205 LET M(I+1)=M(I)
210 NEXT I
215 LET N$(I+1)=T$
220 LET M(I+1)=T
225 NEXT J
230 LET N$(1,21 TO 22)="A+"
235 LET K=1
240 FOR I=2 TO N
245 IF I>K*N/G AND M(I)<>M(I-1) THEN LET
      K=K+1
250 LET N$(I,21 TO 22)=G$(2*K-1 TO 2*K)
255 NEXT I
260 CLS
265 SLOW
270 PRINT AT 4,0;"PRESS";AT 6,0;"'"C'" _FOR_
      PRINTED_COPY";AT 7,0;"'"D'" _TO_
      DISPLAY_RESULTS";AT 8,0;"'"M'" _TO_
      RETURN_TO_COMMAND_MODE";AT 9,0;
      "'"R'" _TO_RE-RUN_PROGRAM";
      AT 10,0;"'"S'" _TO_SAVE_RESULTS"
275 PAUSE 40000
280 LET Q$=INKEY$
285 CLS
290 IF Q$="M" THEN STOP
295 IF Q$="R" THEN RUN
300 IF Q$="S" THEN SAVE "EXAM"
305 IF Q$<>"D" THEN GOTO 365
310 FOR I=1 TO N
315 PRINT I;TAB 3;N$(I);TAB 27;M(I)

```

```
320 IF I < > 15*INT (I/15) THEN GOTO 345
325 PRINT AT 18,0;"PRESS_ENTER_ _ _";AT
    20,0;"OR_ENTER_ANY_CHARACTER_TO_
    RETURNTO_MENU"
330 INPUT Q$
335 IF Q$ < > "" THEN GOTO 260
340 CLS
345 NEXT I
350 PRINT ,, TAB 7;"**_ _**_ _**_ _**"
355 INPUT Q$
360 GOTO 260
365 IF Q$ < > "C" THEN GOTO 260
370 FOR I=1 TO N
375 LPRINT I;TAB 3;N$(I);TAB 27;M(I)
380 NEXT I
385 GOTO 260
9999 FAST
```

APPENDIXES

Speeding Up Programs

Several of the methods used to save space on the T/S 1000 slow down a program, and you may well find it necessary at times to sacrifice memory usage in the interests of maximum speed.

The difference in run times for some of the programs which follow are marginal and may not be noticed in a short program. However, when you have a long program or routine with many loops, you may be surprised to see how much faster a program runs when written for speed. The following suggestions refer mainly to programs which you'll run in **SLOW** mode. In general, **FAST** mode does not need such attention, but there will be cases where it can be significant.

Layout

The most important thing is to ensure that your program listing is well laid out. Have parts of the program which are used most often at the beginning of the listing, and the parts used least often—like the instructions—at the end. There is a simple reason for this. Commands like **GOTO** and **GOSUB** have to search through the program listing to find the line number to jump to. Therefore, subroutines and loops at the start of your programs will be found and executed more quickly than those at the end. To allow you to assign variables and **PRINT** instructions at the start of the **RUN** of a program, make the first line something like **GOSUB 9000**, and then place the initialization procedures and instructions after 9000.

PRINT Statements

Print characters directly in quotes if you can. Time how long these programs take to run:

```
10 FOR F=1 TO 350
20 PRINT "0";"0";
30 NEXT F
```

```
10 FOR F=1 TO 350
20 PRINT 0;0;
30 NEXT F
```

```
10 FOR F=1 TO 350
20 PRINT CHR$ 28;CHR$ 28;
30 NEXT F
```

They should take about nine seconds, eleven seconds and thirteen seconds respectively. The difference between the three runs is so marked you should be able to actually see it as the program runs.

Joining PRINT Statements

You may know you can have any combination of **PRINT/TAB/AT** on any one line of program. However, you may not know that programs run faster if the **PRINT** statements are "chained" than if they are a series of separate program lines. To demonstrate this, **RUN** and time the following two routines:

```
10 FOR N=1 TO 100
20 PRINT AT 10,11;"THINKING"
30 PRINT AT 10,11;"THINKING"
40 PRINT AT 10,11;"THINKING"
```

```

50 PRINT AT 10,11;"THINKING"
60 NEXT N

10 FOR N=1 TO 100
20 PRINT AT 10,11;"THINKING";AT 10,11;
   "THINKING";AT 10,11;
   "THINKING";AT 10,11;"THINKING"
30 NEXT N

```

Logical and Relational Operators

These can be very slow to evaluate. Anything which evaluates to zero takes less time to evaluate than something which evaluates to one. That is, a *false* statement is executed more quickly than is a *true* statement. And it does not stop there. An expression such as **IF X THEN . . .** is quicker than **IF X=1 THEN . . .** and **IF NOT X THEN . . .** is quicker than **IF X=0 THEN . . .**

You will find that **IF X<>1 THEN . . .** evaluates slightly more quickly than does **IF NOT X = 1 THEN . . .**, although the difference is negligible.

Try to arrange your conditional statements so that they usually work out false, and only in a minority of cases does something get executed as a result of a conditional statement. Obviously, if by rearranging a program to take this into account you introduce other factors which slow it down, you've wasted your time and energy, but done carefully it can noticeably speed up programs.

Storing Pictures On Tape

If you have a program which has a complex graphics display—such as sine waves or ellipses—then it may be worthwhile

running the program to produce the picture, and have a line to **SAVE** the program on tape and **RUN** it (a 'load and go' routine) which does not clear the screen. The routine to draw a complex display with such things as sine waves can be very slow, which can be annoying when running a program from scratch. When re-loaded from tape, the picture appears instantly.

2

Enlarged Characters

The T/S 1000 ROM holds a table of data which is used for generating the characters on the display. This table, called the character generator, starts at address 7680 (decimal) and ends at 8191 (decimal). The table contains a series of zeros and ones which correspond to white and black parts of the screen for the characters which have codes in the range 0 to 63 as all others are made up from these characters. Each character square on screen is an 8×8 square. The table consists of eight bytes, each of eight bytes, for each character. Let us take the example of the number 0. 0 has a **CODE** of 28 and the dot patterns start at address 7904. Here is how to work out where the dot patterns for any character start:

$$7680 + (\text{CODE} * 8)$$

and end seven bytes further on at

$$7680 + (\text{CODE} * 8) + 7$$

If we examine the patterns of zeros and ones at these locations in the character generator table we get something like this:

7904	00000000
7905	00111100
7906	01000110
7907	01001010
7908	01010010
7909	01100010
7910	00111100
7911	00000000

If you look at the pattern of ones and imagine them without the zeros then you get something like this:

```

7904
7905   1111
7906   1  11
7907   1  1 1
7908   1 1  1
7909   11  1
7910   1111
7911

```

which looks vaguely like it is seen normally on screen. The way to find out which bits are set and which are not is to repeatedly divide by two, then the remainder tells us if a bit is set (1) or not (0).

Here is a program which uses the **PLOT** facility to create characters four times normal size. When you run the program, enter one character at a time when the **INPUT** quotes appear, followed by **ENTER**. The characters are plotted in the top left corner of the screen, one at a time. The program runs in 1K of RAM memory.

Enlarged Characters



```

30  INPUT A$
40  CLS
50  FOR A=0 TO 7
60  LET P=PEEK (7680+8*CODE A$+A)
70  FOR B=0 TO 7
80  IF P-2*INT (P/2) THEN PLOT 7-B,43-A
90  LET P=INT (P/2)

```

```

100 NEXT B
110 NEXT A
120 GOTO 30

```

This version of the program only allows you to have one enlarged character on the screen at any one time. This was done to simplify the program. The next version introduces the variables **DOWN** and **ACROSS** which control where each character is plotted on the screen. This version only allows characters with **CODEs** in the range 0 to 63.

Large Characters Using PLOT

```

1 2 3 4 5 6 7 8
9 0 1 2 3 4 5 6 7 8
A B C D E F G H I J K L M N O P Q R S
U V W X Y Z [ \ ] ^ _ ` { | } ~

```

```

10 LET ACROSS=7
20 LET DOWN=43
30 INPUT A$
40 LET C=CODE A$
50 FOR K=0 TO 7
60 LET P=PEEK (7680+C*8+K)
70 FOR F=0 TO 7

```

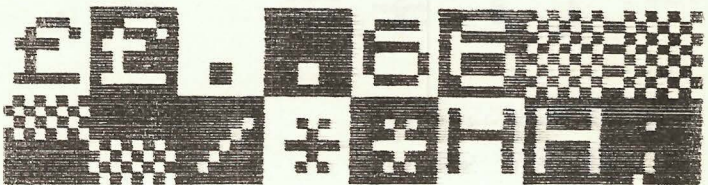
```

80  IF P-2*INT (P/2) THEN PLOT
    ACROSS-F,DOWN-K
90  LET P=INT (P/2)
100 NEXT F
110 NEXT K
120 LET ACROSS=ACROSS+8
130 IF ACROSS>63 THEN LET DOWN=DOWN-8
135 IF ACROSS>63 THEN LET ACROSS=7
140 GOTO 30

```

Large Characters in Inverse Video Using PLOT

This next version permits you to **PLOT** large characters using characters in the range of **CODEs** 0 to 63 plus their inverses. All keywords/functions etc., can be made up of combinations of these. The screen can accommodate up to five rows of eight enlarged characters, a total of 40 characters. This is one advantage of using **PLOT** rather than **PRINT AT**. It means that since the characters are only half the size of those generated by the program using **PRINT AT** more may be accommodated on screen and look neater.



```

10  LET ACROSS=7
20  LET DOWN=43
30  INPUT A$

```

```

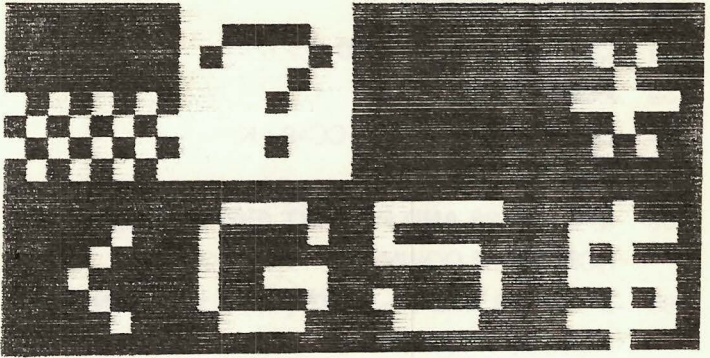
40 LET C=CODE A$
45 LET CC=C-(128 AND C > 128)
50 FOR K=0 TO 7
60 LET P=PEEK (7680+CC*8+K)
70 FOR F=0 TO 7
80 IF C >= 128 AND P-2*INT (P/2)=0 THEN PLOT
   ACROSS-F,DOWN-K
85 IF C < 128 AND P-2*INT (P/2)<>0 THEN PLOT
   ACROSS-F,DOWN-K
90 LET P=INT (P/2)
100 NEXT F
110 NEXT K
120 LET ACROSS=ACROSS+8
130 IF ACROSS > 63 THEN LET DOWN=DOWN-8
135 IF ACROSS > 63 THEN LET ACROSS=7
140 GOTO 30

```

In the above program an extra variable, CC has been introduced. C is the **CODE** of the character to be enlarged and CC is always the **CODE** of that character reduced to a number in the range 0 to 63 for looking up in the character generator. If C is greater than 127, making it an inverse character, this is used in lines 80 and 85 to determine which areas are white and which are black.

Enlarged Characters Using PRINT AT

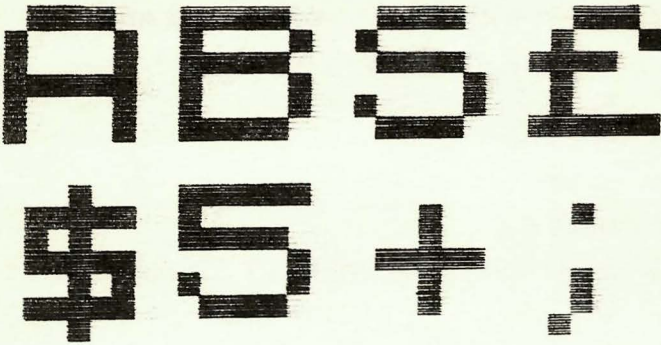
Let us now use **PRINT AT**. Although it has the disadvantages mentioned above, it allows you to enlarge characters larger than **PLOT** and as we'll see, it will allow you a choice of characters other than black on white or white on black. One important point to note is the Y coordinates for **PLOT** and **PRINT AT** work in the opposite directions.



```

10 LET ACROSS=0
20 LET DOWN=0
30 INPUT A$
40 LET C=CODE A$
45 LET CC=C-(128 AND C >= 128)
50 FOR K=0 TO 7
60 LET P=PEEK (7680+CC*8+K)
70 FOR F=0 TO 7
80 IF P-2*INT (P/2) <> 0 AND C < 128 THEN PRINT
   AT DOWN+K,ACROSS+7-F;"■"
90 IF P-2*INT (P/2)=0 AND C >= 128 THEN PRINT
   AT DOWN+K,ACROSS+7-F;"■"
100 LET P=INT (P/2)
110 NEXT F
120 NEXT K
130 IF ACROSS+8 > 31 THEN LET DOWN=DOWN+8
140 LET ACROSS=ACROSS+8 AND
   ACROSS+8 <= 31
150 GOTO 30

```



Enlarged Characters with Tone

And finally here is a version that allows you to print in black on grey (or vice-versa for inverses). This does not work too well on the printer but is great on a TV screen. To use different characters change the **PRINT** statements in lines 80 and 90. Try writing a version that will enable you to select which characters to use for characters and for background. (Hint: use a string.)

```

10 LET ACROSS=0
20 LET DOWN=0
30 INPUT A$
40 LET C=CODE A$
45 LET CC=C-(128 AND C >= 128)
50 FOR K=0 TO 7
60 LET P=PEEK (7680+CC*8+K)
70 FOR F=0 TO 7
75 LET PP=P-2*INT (P/2)
80 IF (PP <> 0 AND C < 128) OR (PP=0 AND
C >= 128) THEN PRINT AT DOWN+K,
ACROSS+7-F;"■"

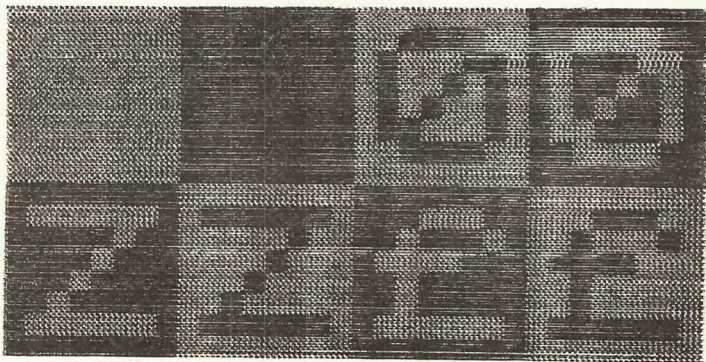
```



```

90  IF (PP=0 AND C<128) OR (PP<>0 AND
    C>=128) THEN PRINT AT DOWN+K,
    ACROSS+7-F;"█"
100  LET P=INT (P/2)
110  NEXT F
120  NEXT K
130  IF ACROSS+8>31 THEN LET DOWN=DOWN+8
140  LET ACROSS=ACROSS+8 AND
    ACROSS+8<=31
150  GOTO 30

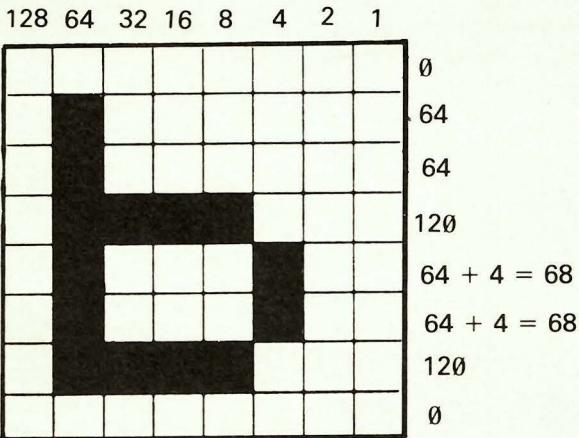
```



Remember, you need not be limited to those characters already in the ROM character. As long as they take the same format (i.e., 8 bytes per character) and you know where they start there is no reason why you could not design an alphabet of lower case letters and store them in a **REM** statement at the start of the program and **PEEK** into this to find the data required to enlarge the letters. Think of the satisfaction when you show your computing friends that your unmodified T/S 1000 can display lower-case letters.

You will find that a piece of graph paper or square ruled paper is almost essential. To design each character, use an 8

× 8 grid as shown below. Fill in those parts corresponding to the character required. For each row, add up the numbers above each shaded area to give a number for each row, from the top down. This will give eight numbers. Follow the example.



You now have a series of numbers.

Now **POKE** these eight numbers into a **REM** statement long enough to hold them and enter the demonstration listing. This has been greatly simplified for the purpose of demonstration. Study the earlier examples if you wish to expand on the routine. Here is the listing:



```

1  REM RND RND????
10 FOR A=0 TO 7
20 LET P=PEEK (16514+A)
30 FOR B=0 TO 7

```

154 / Appendix 2

```
40 IF P-2*INT (P/2) THEN PLOT 7-B,43-A  
50 LET P=INT (P/2)  
60 NEXT B  
70 NEXT A
```

This will give you a lower case b four times normal size. Experiment with the program and expand it to include more characters using earlier examples to help you build up the program.

Useful System Variables

System variables are the part of RAM from 16384 to 16508. They tell the computer facts about itself, such as where the display file starts, which program line should appear at the top of an automatic listing or where the variables area lies in RAM. You can make use of this information in your programs, or you can alter the values contained in some of them to make the T/S 1000 do certain things, such as timing. You can read the values of any of them using **PEEK**, although not all of them are particularly useful for the programmer.

Address	Notes
16384	Controls error report code. If you POKE one of these values into 16384 the program stops without displaying an error code: 43,70,72, 73,74,75,76,77,79,81,82 or 89. The program is now in command mode: the next key pressed is interpreted as a keyword.
16388/16389	The address of the first byte above the BASIC system area, sometimes known as RAMTOP. NEW only operates up to the address held in RAMTOP, so anything placed above this is not wiped from memory. If this value is less

(Continued)

Address	Notes
	than 19712 then when CLS is executed, the display file is contracted to its minimum size.
16396/16397	The address of the beginning of the display file is contained in this system variable. Should not be POKEd . You can also find the length of a program listing (i.e., less variables, display, etc.) with this.
16398/16399	Address of PRINT position in display file. Can be POKEd so that the PRINT position is sent elsewhere.
16400/16401	Address of the start of the variables area. Useful for machine code programs where you might want to transfer data held in strings/arrays.
16419/16420	The number of the top program line in automatic listings. May be changed so that a different line goes on top.
16425	Address of next program line to be executed. Useful if you want to POKE into a line of BASIC in the middle of a program.
16434	The seed for RND . Useful for games involving codes and ciphers generated by RND/RAND since this is the variable set by RAND .
16436/16437	Frame counter. Bit 15 is 1. Bits 0 to 14 are decremented for each frame sent to the TV down to a minimum of 32768, because of BIT 15. PAUSE resets BIT 15 to 0 and sets the other bits to the length of the pause. The PAUSE ends when zero is reached. If the PAUSE ends early then BIT 15 is usually set to 1 again.
16438	X – coordinate of last point plotted.
16439	Y – coordinate of last point plotted.

(Continued)

Address	Notes
6441	Column number of PRINT position on screen.
16442	Line number of PRINT position. You can PEEK this to see if the screen is nearly full and if so, execute CLS .
16444 to 16476	Printer buffer. Consists of the 32 characters of a line plus ENTER . If printer is not used, this area may be used as a temporary store for information.

4

Decimal to Hexadecimal Converter

This is a short program which converts any decimal number from zero to 255 which you enter, into hexadecimal notation. It may be useful for machine code programmers.

```
10 PRINT AT 0,0;"ENTER_A_DECIMAL_NO. (0_
   TO_255)"
20 INPUT X
30 SCROLL
40 IF X < > INT X OR X < 0 OR X > 255 THEN GOTO 10
50 PRINT TAB 3-LEN STR$ X;X;"_DECIMAL_IS_";
60 PRINT CHR$(28+INT(X/16)) + CHR$(28+(X-INT
   (X/16)*16));
70 PRINT "_HEXADECIMAL"
80 GOTO 10
```

Line 10 asks the user for a number in the desired range, and this prompt is placed at the very top of the screen so that when the display scrolls up the prompt is lost until it is time for another input, and the prompt does not appear in the middle of the scrolling list of conversion values. Line 40 rejects unwanted out of range values and non-integral values by sending the program back to the user input section. Line 50 spaces the

number entered tidily so that everything is placed neatly under each other. It does this by examining the length of the number converted to a string using the **STR\$** function. Line 60 does the conversion. You could save memory slightly if you wanted to by combining the three **PRINT** statements in lines 50, 60 and 70 into one long **PRINT** statement.

Hexadecimal to Decimal Converter

This is a 1K program that converts any hexadecimal value from 00 to FF (i.e., single byte) to its equivalent decimal value, the reverse function of the preceding program. You may like to join the two programs with an option to switch from one function to the other.

```

10  PRINT AT 0,0;"ENTER _A_ HEXADECIMAL _
    NO.00 _TO_ FF"
20  DIM X$(2)
30  INPUT X$
40  SCROLL
50  IF X$(1)<"0" OR X$(1)>"F" OR X$(2)<"0" OR
    X$(2)>"F" THEN GOTO 10
60  PRINT X$;" _HEX_ IS _";
70  LET X=(CODE X$-28)*16+CODE X$(2)-28
80  PRINT TAB 14-LEN STR$ X;X;" _DECIMAL"
90  GOTO 10

```























The program is very similar to the previous one; since hex values in the range we are interested in always consist of two characters, e.g. 00 or FF. So line 20 ensures that X\$ (the hex string to be converted) always consists of 2 characters.

Line 50 checks that X\$ characters are permissible hex characters in the range 0 to F. Line 50 prints the first part of the conversion and here there is no need for a special spacing method as before because the hex value will always be the same length and space itself out neatly. Line 70 converts the hex characters to a decimal number from 0 to 255. This will require spacing out, and this is done in line 80 using a similar method to the previous program.

6

Special Graphics Symbols

The following graphics symbols are available on the T/S. Please note that in this listing each symbol is "boxed," meaning that all four of its borders are shown even though these borders will *not* appear on the screen. For example, the very first symbol listed here will show up on the screen merely as a space, and not an empty box, and in listings in this book is indicated as an underscore (). It should also be noted that letters in inverse video are also accessed via the **G** mode.

Symbol	Mode	Key	Symbol	Mode	Key
	K or L	SPACE		G	SPACE
	G	shifted-1		G	shifted-Q
	G	shifted-2		G	shifted-W
	G	shifted-7		G	shifted-6
	G	shifted-4		G	shifted-R
	G	shifted-5		G	shifted-8
	G	shifted-T		G	shifted-Y
	G	shifted-E		G	shifted-3
	G	shifted-A		G	shifted-H
	G	shifted-D		G	shifted-G
	G	shifted-S		G	shifted-F

BOGGLERS

22

Smart Games

Programs

(2K to 16K)

in

Timex/Sinclair BASIC

Graham Charlton • Mark Harrison • Dilwyn Jones

Edited by Yin Chiu

**BACKGAMMON ■ QUATERMASS ■ CHECKERS ■ SUB HUNT ■
GOMOKU ■ ELIZA ■ BIORHYTHMS ■ FOUR IN A ROW ■
OTHELLO/REVERSI ■ FOX AND GEESE ■ PAIRS ■ FOUR FIELD
KONO ■ SURF RIDER ■ HIGH JUMP ■ MAZE MASTER ■
ZOMBIES ■ POKER DICE ■ ANTIMIND ■ NIM ■ HUNDRED UP
■ TIC-TAC-TOE ■ SOLITAIRE ■**

Here are twenty-two entertaining and intelligent games programs in Timex/Sinclair BASIC. They can be run on the T/S 1000 with a 16K-RAM (random access memory) expansion module as well as on any other Timex machine with 16K RAM memory that uses Timex BASIC. All of the programs have been fully tested, and once entered into your computer, they will provide many hours of entertainment and instruction for all members of the family. Try them all.

Graham Charlton, a frequent contributor to numerous British computer magazines, spends a good deal of time working with the Sinclair computers.

Mark Harrison is a student and the author of *Byteing Deeper Into Your ZX81*.

Dilwyn Jones is coauthor of *Programming Your ZX SPECTRUM* and *Mastering The Timex/Sinclair 1000*.

Yin Chiu is coauthor of *CRUNCHERS* and works as a programmer with a large New York bank.



ISBN 0-07-023959-2