COMPUTER CHALLENGES FOR THE AMSTRAD

Richard Hurley David Virgo

COMPUTER CHALLENGES FOR THE AMSTRAD

Computer Challenges for the Amstrad

Richard Hurley
& David Virgo

Duckworth

First published in 1985 by
Gerald Duckworth & Co. Ltd.
The Old Piano Factory
43 Gloucester Crescent, London NW1

ISBN 0 7156 1979 9

British Library Cataloguing in Publication Data

Hurley, Richard G.
Computer Challenges for the Amstrad.
1. Computer games 2. Amstrad computer
— Programming
I. Title II. Virgo, David
794.8'028'5404 GV1469.2

ISBN 0-7156-1979-9

Photoset in North Wales by
Derek Doyle & Associates, Mold, Clwyd
Printed in Great Britain by
Redwood Burn Ltd., Trowbridge

Contents

This book is dedicated to my mother,
Mrs Elizabeth Virgo, for all her love
and encouragement over the years.

Acknowledgments

We would like to express our sincere thanks to Stephen Lacey, who has contributed extensively to the design of the programs and the formulation of the text. We are also grateful to Josie for postponing her annual holiday in order that this book might be completed on time.

Introduction

Computer Challenges is a book of nine programs with hints and advice on Locomotive Basic and instructions on the use of artificial intelligence in programming. There are four puzzles and five games, and you will need to exhibit clear thinking and much skill if you are to solve the puzzles and beat the computer at the games it offers.

The first two chapters introduce you to some of the principles of artificial intelligence and the more advanced features of Locomotive Basic which are used in this book. The rest of the book is divided into two parts: the first contains puzzles for you to solve; the second a number of intellectually stimulating games in which the computer will be a worthy adversary.

As well as nine worthwhile listings, we include a large quantity of information on the development of each program together with detailed documentation of its operation. The final chapter is a review of the application of artificial intelligence in the above programs, with some detailed notes on the creation of another game, offering you the opportunity to write a program of your own.

Each listing contained in this book is preceded by a warning such as that shown below. It must be strictly adhered to.

The game should be entered and played in the caps lock mode.
All spaces within the text should be entered as they appear.

Before typing any of the programs into the Amstrad, you should familiarise yourself with the operation of the Datacorder and its associated commands, ensuring that two or more quality cassettes are available for storage. We recommend that you keep two copies of each program on separate tapes at all times, in case any problems are encountered during program transfer. We also recommend that you reset the Amstrad before any program is loaded in order to restore any preset windows and colours.

1. Artificial Intelligence

It is easy to remark glibly that the programs contained in this book demonstrate the use of artificial intelligence, but to the average reader this will mean little unless we can provide a satisfactory definition. There is little doubt that trying to define artificial intelligence is as difficult, if not more so, than using the principles in our programs. We will therefore consider the following simple definition:

> Artificial intelligence is the combination of methods employed in programming in order to make a computer solve or attempt to solve a problem by selective reasoning.

In this book we will employ several artificial intelligence techniques to provide you with nine puzzles and games which will prove to be entertaining as well as instructive. As all these programs are written to run on the Amstrad, we will restrict ourselves to the use of Basic which will make the listings easy to follow and understand.

Heuristics

It is common knowledge that a program generally consists of a series of lines (instructions) which, when executed in sequence, give the solution to a problem. This group of instructions is usually defined as an 'algorithm', and the art of solving a problem using a computer usually comes down to finding some suitable algorithm for the task.

This approach fails in a case such as Draughts (Chapter 11), however, where the problem is to find the best possible move at any time during the game. It is not possible to evaluate this exactly, and under such circumstances we employ a 'heuristic', which can be defined as a rule or set of rules which, when applied, lead us closer to the solution. Heuristics are not able to guarantee a solution to a problem, but they can be used in a wide range of

applications including games and knowledge-based expert systems.

Games

Consider the problem of writing a games program such as Draughts, in which the computer is to play a reasonable game. We need to define a set of rules which, when applied at the time of the computer's turn, will produce a move resulting in a situation closer to the solution. The solution is defined as a board position which contains none of the opponent's pieces.

Under these circumstances we might use rules such as:

```
if piece is captured then move   = very good
if piece is crowned them move    = good
if position is attack then move  = fair
```

Each of these rules must then be converted into a form which can be represented mathematically and the relative difference between the adjectives 'good' and 'fair' must be evaluated, giving rise to instructions such as:

```
IF PC=1 THEN LET W=W+10
IF K=1 THEN LET W=W+8
IF PA=1 THEN LET W=W+5
```

Where W is the weighting given to each move, and PC, K and PA are flags set to 1 when a particular situation arises.

Therefore, to write a program in which the computer plays a good game we must formulate a very comprehensive set of rules and apply these to every possible move.

Minimax procedure

The method described above will satisfactorily find the best possible move for the computer, but it does not take the opponent's response into account, which in any game involving two players is of paramount importance. When it is the computer's turn to move it must find the move which gives it the maximum benefit. However, if the result of this move is to give the opponent an even better position, then this original maximum is not as good as was first thought.

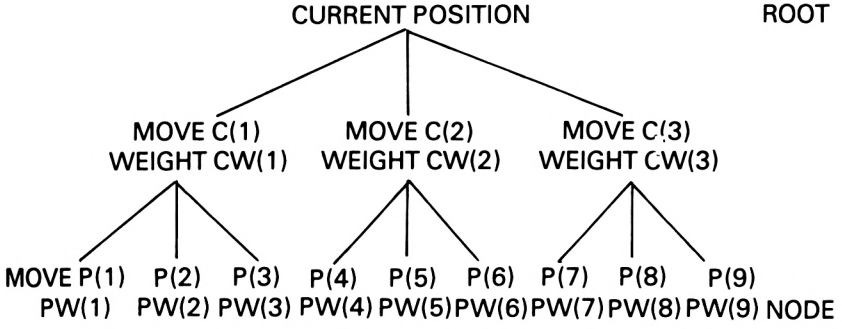
This problem can only be overcome by using the 'minimax

procedure' in which we consider the *maximum* move for the computer which results in the *minimum* move for the opponent. This method, however, has a profound effect on the time taken for each move, since for each of the computer's moves, all of the opponent's responses must be considered.

Look-ahead

We see that for a computer to play any game well against an opponent, it is necessary to consider several moves into the future. This can be very time-consuming, especially in the case of Draughts where the number of combinations during the middle game can be astronomical, and to consider each of them would be totally impractical.

For example, it is possible to consider all of the moves at any stage of a game by looking at the 'move-tree', and a section of a Draughts move-tree is shown below. Note that only a section of the tree has been constructed, as to show every branch would take far too much space.



This tree represents a position in which the computer and the player have only three legal moves each. In practice this is a very unlikely situation! To evaluate the best move we could scan down the tree finding the values of CW(1), CW(2) and CW(3) for the computer's move and then subtracting the values of PW(1), PW(2) and PW(3), etc., representing the weight of the player's moves. When all the possible combinations have been considered we select the move which results in the greatest value of CW–PW.

In practice it is likely that we will need to search to a depth of at least three moves, and assuming a realistic number of legal moves, for instance twelve for both the computer and the player,

we would have 1728 possible combinations to consider. Hence the method of scanning the tree from the root to each possible node would be very time-consuming.

If we are using a 'minimax' heuristic, however, it becomes a simple matter to prune the tree. This is achieved by starting at the root and scanning horizontally as opposed to down to the nodes. In the above example the whole of level 1 would be scanned to find the maximum weighted move. The second level would then be considered for the best move to find the final weight, and thereafter the second level would only be considered in cases where the level 1 weighting is greater than the previous final value. This method works well in practice and, as can be seen in Draughts, produces a program which plays a challenging if not fool-proof game.

Throughout this book we will consider the heuristic as opposed to the algorithmic approach in writing games programs. It should be appreciated that this approach to programming not only applies to games, but any field in which an algorithm cannot be found to produce the solution.

2. Locomotive Basic

Locomotive Basic is a very versatile, fast and powerful version of Basic, and like all versions, it contains some peculiar commands and additional facilities. This chapter will highlight some of those that have been employed in the programs in this book, considering the subjects of cassette file management, interrupts and graphical procedures in some detail.

The following pages, in conjunction with the manual, will give you enough information to unleash your imagination on the design of more efficient and sophisticted programs.

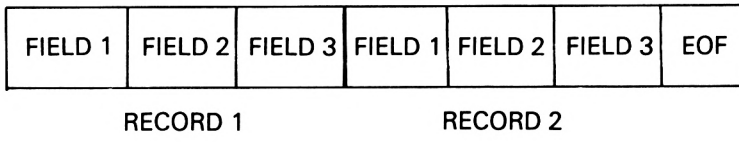
File handling

One of the major advantages of a computer is its ability to store large quantities of information, which it can then search, interrogate and display at high speed. The method of dealing with information in this way is generally referred to as 'data processing'. In this section we will consider how this can be achieved using an Amstrad computer and the Datacorder to store the information.

Files

In data processing, the quantity of information being considered is often very large. It is therefore impractical to store all the variables in data statements or in arrays within the programs. Under such circumstances, the data would be stored in a file which could be extracted from the backing store (Datacorder) under direct program control.

File structure



A file is divided up into a series of records with the individual records being divided into a number of fields. Each field can be numeric, alphabetic or alphanumeric and can contain one item of data (e.g. a name, number, score, etc.)

When handling a file, it is imperative to remember its structure, paying special attention to:

1. The number of fields per record
2. The type of each field
3. Which field represents the key

Key field: in general, the first field of each record is called the key field. It is used to identify a particular record, separating it from all the others in the file. In most cases, the key field will be a name, but this is not necessarily the case, since some form of code could be used to protect the integrity of the data.

End of file (EOF): when the computer is required to read information contained within a file, it is most unlikely that it will have prior knowledge of the number of records that the file contains. If under these circumstances we attempt to read the file by using a FOR NEXT loop, then it is highly likely that a situation will arise in which the computer is attempting to extract records from a file that contains no more data. To overcome this problem, the Amstrad puts an 'end of file' (EOF) marker on to the cassette when data transmission has been completed. The EOF marker is a Boolean-type variable, and during the reading stage the end of the file can be tested by using a program line such as:

```
100 IF EOF THEN GOTO 150
```

Commands

There are seven commands that relate to cassette handling on the Amstrad. They will be described in this chapter and used throughout the book. The commands are as follows:

OPENIN	OPENOUT
CLOSEIN	CLOSEOUT
EOF	PRINT #9
INPUT #9	

OPENOUT "TEST"
This is the first command to be used when writing a file on to tape. Information can be transferred from memory to the Datacorder via a 2K buffer. The data will not be transferred from the buffer until either it is full or a CLOSEOUT command is encountered.

OPENIN "TEST"
This is the same as the OPENOUT command except that data will flow in the opposite direction, i.e. from the tape into the computer's memory. As with the output mode, data will be transferred via the buffer, with the actual transmission occurring when the buffer is full or when a CLOSEIN command is encountered.

Note that if the first character of a file name is an exclamation mark then the cassette processing messages normally displayed on the screen will be suppressed, leaving what appears on the screen to the programmer's discretion.

CLOSEOUT
When the computer has finished writing a file, the file must be closed by using the CLOSEOUT command, which transfers any information remaining in the input/output buffer and places the all-important EOF marker at the end of the file.

CLOSEIN
This is similar to the CLOSEOUT command, but is used to close an input file. Note that when the computer encounters an OPEN or CLOSE command a large quantity of internal memory management is carried out and this can cause quite substantial delays (see Chapter 5).

PRINT #9
This command is used to transfer data to the Datacorder. Its effect is to send one item of data or one field, which is then stored on tape.

INPUT #9
This is the opposite of the PRINT# command, transferring one item of data (field) from the file into the main store via the input/output buffer.

Types of data-processing programs

There are three distinct types of data-processing or file-handling programs:

1. Write only
2. Read only
3. Read and write

As the names suggest, the first type is used to construct an original file, the second to interrogate or read an existing file, and the third is able to read, change and then re-write the file back on to tape.

Write only

In this category of program, the object is to create a file on tape where none previously existed. To achieve this, data must first be collected using LET, READ or INPUT instructions, and transferred to a file using the PRINT #9 statement.

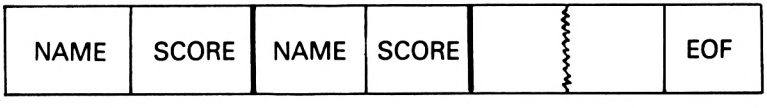
Example 1
This program constructs a file called SCORE, which contains ten names and the scores obtained in a game. It is assumed that when the original file is created no one has played the game and a set of dummy names and scores are used when initialising the file.

```
10 REM **FILE CREATE**
20 CLS
30 DIM N$(10),S(10)
40 FOR I=1 TO 10
50 LET N$(I)="AMSTRAD"
60 LET S(I)=100
70 NEXT I
80 OPENOUT "SCORE"
90 FOR I=1 TO 10
100 PRINT#9,N$(I)
110 PRINT#9,S(I)
120 NEXT I
130 CLOSEOUT
140 END
```

Comments

Lines 10-70	Initialise variables
Line 80	Open file
Lines 90-120	Transfer records from memory to tape
Lines 130-140	Close file and end program

When this program has finished, a file called SCORE with a structure as indicated below will exist on tape.



RECORD 1 RECORD 2

Read only

Having a created a file on tape using a write only program, as demonstrated above in Example 1, there will come a time when it is necessary to display the contents on the screen. This can be achieved by using a read only program as shown below.

Example 2
This program reads the SCORE file created in Example 1 and prints the score table on the screen.

```
10 REM **FILE READ**
20 CLS
30 DIM N$(10),S(10)
40 OPENIN "SCORE"
50 FOR I=1 TO 10
60 INPUT#9,N$(I)
70 INPUT#9,S(I)
80 NEXT I
90 CLOSEIN
100 CLS
110 LOCATE 5,1:PRINT "HIGH SCORES"
120 LOCATE 5,2:PRINT "---- ------"
130 PRINT "NAME","SCORE"
140 PRINT "----","-----"
150 FOR I=1 TO 10
160 PRINT N$(I),S(I)
170 NEXT I
180 END
```

Comments

Lines 10-30	Initialise variables
Line 40	Open file
Lines 50-80	Transfer data to CPU
Line 90	Close file
Lines 100-180	Display score table on the screen

Read and write

The third and most useful type of file-handling program is a read and write or file management program. As its name suggests, the program assumes that a file already exists, which can then be loaded into the computer, interrogated, changed and then written back on to the tape.

Example 3
This is a file management program for handling the SCORE data file created in Example 1.

```
10 REM **FILE MANAGEMENT**
20 CLS
30 DIM N$(10),S(10)
40 OPENIN "SCORE"
50 FOR I=1 TO 10
60 INPUT#9,N$(I)
70 INPUT#9,S(I)
80 NEXT I
90 CLOSEIN
100 CLS
```

```
110 LOCATE 5,1:PRINT "MENU"
120 LOCATE 5,2:PRINT "----"
130 PRINT
140 PRINT "1....DISPLAY TABLE"
150 PRINT "2....INPUT NEW SCORE"
160 PRINT "3....SAVE FILE"
170 PRINT "4....END"
180 PRINT:PRINT
190 INPUT "SELECTION=";Z
200 IF Z<1 OR Z>4 THEN GOTO 130
210 ON Z GOTO 220,350,540,620
220 REM **DISPLAY**
230 CLS
240 LOCATE 5,1:PRINT "HIGH SCORES"
250 LOCATE 5,2:PRINT "---- ------"
260 PRINT "NAME","SCORE"
270 PRINT "----","-----"
280 FOR I=1 TO 10
290 PRINT N$(I),S(I)
300 NEXT I
310 PRINT
320 PRINT "PRESS ENTER TO CONTINUE"
330 IF INKEY$="" THEN GOTO 330
340 GOTO 100
350 REM **NEW SCORES**
360 CLS
370 INPUT "NEW SCORE=";NS
380 IF NS>S(10) THEN GOTO 430
390 LOCATE 5,5:PRINT "SCORE DOES NOT REGISTER"
400 LOCATE 5,7:PRINT "PRESS ENTER TO CONTINUE";Z$
410 IF INKEY$="" THEN GOTO 410
420 GOTO 100
430 INPUT "PLAYERS NAME=";NN$
440 FOR I=1 TO 10
450 IF NS>S(I) THEN GOTO 470
460 NEXT I
470 FOR J=10 TO I+1 STEP -1
480 LET S(J)=S(J-1)
490 LET N$(J)=N$(J-1)
500 NEXT J
510 LET N$(I)=NN$
520 LET S(I)=NS
530 GOTO 100
540 REM **SAVE FILE**
550 OPENOUT "SCORE"
560 FOR I=1 TO 10
570 PRINT#9,N$(I)
580 PRINT#9,S(I)
590 NEXT I
600 CLOSEOUT
610 GOTO 100
620 END
```

Comments

Lines 10-30	Initialise variables
Lines 40-90	Load file into memory
Lines 100-210	Display and select from menu
Lines 220-340	Print score table on the screen
Lines 350-530	Input new score and position in table if appropriate
Lines 540-610	Save new file on to cassette tape
Line 620	End program

Variables

The table shown below lists the main variables used in Examples 1, 2 and 3. This may help you to appreciate how the programs operate.

N$()	NAMES
S()	SCORES
NN$	NEW NAME
NS	NEW SCORE

Conclusion

The techniques discussed here are extremely important in many branches of programming. To perform well and in an intelligent manner, a computer must have access to large quantities of information. However, due to the limitations set by the size of the computer's RAM this information cannot be stored within the program itself, and the programmer must resort to the use of files.

In the Crossword program in Chapter 5 the techniques considered here are put to practical use in the construction of a sophisticated program which relies totally on the information contained within a large data file.

Interrupts

One of the most powerful features of Locomotive Basic is the way in which it allows the user to control interrupts via Basic keywords. An interrupt occurs when the normal execution of a program is 'interrupted' so that some other course of action may be taken.

On large main-frame computers, the execution of a program may be interrupted to allow a more important program to be executed, or to permit a program which requires less CPU time to

have a higher priority. On microcomputers, the execution of a program may be stopped from the keyboard, by either the break key, the escape key, or control-C combination.

On the Amstrad, in addition to the escape key, there is an extremely powerful interrupt-handling system. By using this, it is possible to multi-task two or more routines from within the same program, enabling facilities such as on-screen timers and continuous music to be produced without machine-code routines.

Commands

There are ten commands which deal with the handling of the interrupts:

AFTER x,y GOSUB z
EVERY x,y GOSUB z
REMAIN (x)
ON SQ (x) GOSUB y
ON ERROR GOTO x
ON BREAK GOSUB x
ON BREAK STOP
RESUME
EI
DI

AFTER x,y GOSUB z
The Amstrad has a built-in real-time clock which allows a maximum of four delay-timers to be created.

'x' is an integer variable in the range 0-32767. This is the delay time in 1/50th of a second, after which the routine at line 'z' will be executed. At the end of the routine there must be a RETURN statement, transferring control back to the main program, which continues from where it was interrupted.

'y' gives the number of the timer to be used in the range 0-3. (0 is assumed if no value is given.) The four timers have different priorities, 3 being the highest priority and 0 the lowest, and in the case of more than one interrupt being required simultaneously, the one with the highest priority will be executed first.

Example 1
Using the AFTER command, a program can be written which waits for a period of time to elapse before an alarm is sounded. When executed, the program will ask for the length of delay and the border will then flash until the specified time has elapsed, when the alarm routine at Line 150 will be executed.

```
10 REM **ALARM CLOCK**
20 MODE 1
30 BORDER 0
40 INK 0,0
50 LOCATE 5,12
60 INPUT "HOW LONG TO DELAY (SECS) ";X
70 IF X>655 THEN RUN
80 LET X=X*50
90 AFTER X GOSUB 150
100 FOR I=0 TO 26
110 BORDER I
120 FOR K=1 TO 100:NEXT K
130 NEXT I
140 GOTO 100
150 BORDER 0
160 SOUND 1,100,50
170 BORDER 6
180 SOUND 1,150,50
190 GOTO 150
```

EVERY x,y GOSUB z

This command is essentially the same as the AFTER command except that it will arrange for a routine to be called at regular intervals rather than after a period of time.

Example 2

This program produces an on-screen real-time clock which increments each second and chimes on the minute.

```
10 REM **CHIMING CLOCK**
20 MODE 1
30 ENV 1,5,-3,10
40 INK 0,0
50 INK 1,24
60 INK 2,26
70 INK 3,6
80 WINDOW#2,18,23,10,15
90 PAPER#2,2
100 PEN#2,3
110 CLS#2
120 LOCATE#2,2,2
130 PRINT#2,"TIME"
140 LOCATE#2,2,4
150 PRINT#2,"0:00"
160 LET MI=0
170 LET SEC=0
180 EVERY 50 GOSUB 270
190 EVERY 3000,1 GOSUB 370
200 LET X=INT(RND*26)
210 INK 0,X
```

```
220 LET Y=INT(RND*26)
230 BORDER Y
240 FOR I=1 TO 200
250 NEXT I
260 GOTO 200
270 LET SEC=SEC+1
280 IF SEC=60 THEN LET SEC=0:LET MI=MI+1
290 IF SEC=0 AND MI=10 THEN LET MI=0
300 LOCATE#2,2,4
310 PRINT#2,USING "#";MI;
320 PRINT#2,":";
330 PRINT#2,USING "##";SEC
340 IF SEC<10 THEN LOCATE#2,4,4:PRINT#2,"0"
350 SOUND 1,0,1,7,0,0,7
360 RETURN
370 FOR H=1 TO MI+1
380 SOUND 2,100,0,15,1
390 NEXT H
400 RETURN
```

Comments

Lines 10-170	Initialise program
Lines 180-190	Initialise timers
Lines 200-260	Flash colours
Lines 270-360	Increase clock and produce tick
Lines 370-400	Produce chimes

REMAIN (x)

This command returns the amount of time remaining in the timer 'x' and is used in the form:

PRINT REMAIN (1)

or

LET D = REMAIN (0)

When used, the REMAIN command will return the relevant value and disable the timer that it refers to, unless it refers to a timer which is already disabled, when it will return 0.

Example 3

The object of this program is to find your average reaction time over a series of ten tests. After a short wait the border on the screen will flash and you must press the space bar as quickly as possible. After each test the timer is reset and your running average is printed on the screen.

The program incorporates all three techniques discussed so far

in this section. The AFTER command at line 300 delays the program for a random period of time with a maximum of ten seconds. The border flashes and the clock is started. It is stopped when the space bar is pressed, by using the REMAIN command.

```
10 REM **REACTION TIMER**
20 MODE 1
30 DIM AV(10)
40 BORDER 0
50 INK 0,0
60 INK 1,24
70 INK 2,26
80 INK 3,6
90 WINDOW#2,18,24,2,7
100 WINDOW#3,16,26,10,15
110 PAPER#3,2
120 CLS#3
130 PEN#3,3
140 PAPER#2,2
150 PEN#2,3
160 CLS#2
170 LOCATE#3,3,2
180 PRINT#3,"AVERAGE"
190 LOCATE#3,5,4
200 PRINT#3,"0.0"
210 LET NO=1
220 FOR GO=1 TO 10
230 LET A=0
240 LOCATE#2,2,2
250 PRINT#2,"TIMER"
260 LOCATE#2,2,4
270 PRINT#2," 0.0"
280 LET TE=0
290 LET SEC=0
300 AFTER INT (RND*500) GOSUB 520
310 IF A=0 THEN GOTO 310
320 IF INKEY(47)=0 THEN LET D=REMAIN(0):GOTO 340
330 GOTO 320
340 LET AV(NO)=SEC+(TE*0.1)
350 LET AS=0:FOR I=1 TO NO:LET AS=AS+AV(I):NEXT I
360 LET AS=AS/NO:LOCATE#3,4,4
370 LET AS=INT(AS*100)/100
380 PRINT#3,AS
390 LET NO=NO+1:NEXT GO
400 LOCATE 16,20
410 PRINT"ANOTHER GO?"
420 IF INKEY$="Y" THEN RUN
430 IF INKEY$="N" THEN END
440 GOTO 420
450 LET TE=TE+1:IF TE=10 THEN LET TE=0:LET SEC=SEC
+1
```

```
460 IF SEC=60 AND TE=0 THEN LET SEC=0
470 LOCATE#2,2,4
480 PRINT#2,USING "##";SEC;
490 PRINT#2,".";
500 PRINT#2,USING "#";TE
510 RETURN
520 BORDER 6:SOUND 1,100,10:BORDER 0
530 EVERY 5 GOSUB 450
540 LET A=1
550 RETURN
```

Comments

Lines 10-210	Initialise program
Lines 220-390	Main loop for the ten tests
Lines 390-440	End routine
Lines 450-510	Clock routine
Lines 520-550	Start clock

ON SQ(x) GOSUB y

This command creates an interrupt that will call a routine when there is a free slot in the sound queue 'x' (for notes on sound and queues see the Amstrad manual). When the routine has finished, a RETURN statement will transfer control back to the main program at the point from which it was interrupted.

After an ON SQ GOSUB command has been performed, it will automatically be disabled. If the interrupt is required again, it will have to be reset.

The main use for this command is in the creation of continuous music, as demonstrated in the following program.

Example 4

This program demonstrates how a tune can be played while the computer is performing other calculations.

```
10 REM **CHARIOTS**
20 ON SQ(1) GOSUB 140
30 INK 0,0
40 BORDER 0
50 INK 1,24
60 MODE 1
70 FOR I=1 TO 12
80 LOCATE 10,I*2
90 PRINT I;" * 12 = ";I*12
100 FOR DELAY=1 TO 800
110 NEXT DELAY
120 NEXT I
130 GOTO 60
140 READ CHAN,PITCH,DUR
```

```
150 IF CHAN=-1 THEN RESTORE:GOTO 170
160 SOUND CHAN,PITCH,DUR
170 ON SQ(1) GOSUB 140
180 RETURN
190 DATA 1,225,20,1,169,20,1,150,20,1,134,20
200 DATA 1,150,60,1,179,60
210 DATA 1,225,20,1,169,20,1,150,20,1,134,20
220 DATA 1,150,120
230 DATA 1,225,20,1,169,20,1,150,20,1,134,20
240 DATA 1,150,60,1,179,60
250 SOUND 0,0,60
260 DATA 1,150,20,1,169,20,1,179,20,1,201,20,1,225
,120
270 DATA -1,0,0
```

Comments

Lines 10-50	Initialise interrupt and set colours
Lines 60-130	Display twelve times table
Lines 140-180	Play music
Lines 190-270	Data for music

ON ERROR GOTO x / RESUME

The ON ERROR GOTO x command creates an interrupt which will branch to a separate routine when an error occurs within the main program. When an error is detected values are given to two variables: ERL contains the line number where the error was detected and ERR contains the error number (a full list of error numbers is supplied in Appendix VIII of the Amstrad manual).

When the required routine has finished, a RESUME command will cause the main program to continue from the point at which the error was detected.

ON BREAK GOSUB x

This command initialises an interrupt which will call a routine at line 'x' when the escape key is pressed twice. This can be used to prevent someone breaking into programs and thereby protect them from would-be 'pirates'! For example:

```
 10 ON BREAK GOSUB 100
 20 REM **THE PROGRAM**
100 NEW
```

Example 5

This program demonstrates the use of the ON BREAK GOSUB command. You will be asked to play a little guessing game, but if at any time you wish to stop, press the escape key twice and a message will be displayed on the screen.

```
10 REM **GUESSING GAME**
20 ON BREAK GOSUB 290
30 LET NUM=INT(RND*10)+1
40 CLS
50 LOCATE 11,2
60 PRINT "GUESSING GAME"
70 PRINT
80 PRINT"I HAVE A NUMBER BETWEEN 1 AND 10."
90 FOR I=1 TO 5
100 LOCATE 1,6
110 PRINT"GUESS NO.";I
120 PRINT
130 PRINT"YOUR GUESS ?";
140 LET A$=INKEY$
150 IF A$="" THEN GOTO 140
160 PRINT A$
170 LET A=VAL(A$)
180 IF A<1 OR A>10 THEN GOTO 220
190 IF A=NUM THEN PRINT:PRINT"CORRECT !":GOTO 270
200 IF A<NUM THEN PRINT:PRINT"TOO LOW.":GOTO 220
210 IF A>NUM THEN PRINT:PRINT"TOO HIGH."
220 FOR J=1 TO 800:NEXT J
230 LOCATE 12,8:PRINT "                "
240 LOCATE 1,10:PRINT "          "
250 NEXT I
260 PRINT:PRINT"OUT OF GUESSES."
270 FOR J=1 TO 1600:NEXT J
280 GOTO 30
290 CLS
300 LOCATE 16,12
310 PRINT"GOODBYE."
320 LOCATE 1,24
```

Comments

Line 20	Initialise interrupt
Lines 30-280	Play the game
Lines 290-320	Interrupt routine

ON BREAK STOP

This command disables the ON BREAK GOSUB interrupt. When used inside an ON BREAK GOSUB interrupt routine it will allow the interrupt to work once only, warning the user that he is about to break out of the program.

DI and EI

These commands are abbreviations for Disable Interrupts (DI) and Enable Interrupts (EI).

DI is used to suppress the execution of any routines called by interrupts. This may be useful when printing to the screen is being

performed. EI performs the opposite function and is used to cancel the effect of any previous DI instructions.

When an interrupt routine is called, an automatic DI is executed to prevent the routine from interrupting itself. In a similar way, the RETURN statement at the end of the routine executes an EI statement. Care must be taken not to jump out of an interrupt routine without first enabling the interrupts, otherwise no further interrupts will function!

Example 6
This program demonstrates the power of DI and EI. When the program is executed, the word HELLO will move across the screen and a counter will increase at the bottom of the screen. Now try the program without lines 30 and 70 and you will see a very different result.

```
10 CLS
20 INK 2,15
30 EVERY 10 GOSUB 140
40 FOR I=-50 TO 640
50 DI
60 PLOT 0,0,1
70 MOVE I,200
80 TAG
90 PRINT"HELLO";
100 TAGOFF
110 EI
120 NEXT I
130 END
140 LOCATE 1,25
150 PEN 2
160 PRINT"COUNTER:-";
170 PRINT A
180 LET A=A+1
190 RETURN
```

Conclusion

As can be seen from the examples considered in this section, the control of interrupts opens new areas for Basic programmers, previously only available in machine code.

Graphics

Any computer programmer will eventually need to incorporate some form of graphics in a program. This section will consider the

commands of Locomotive Basic that are available for this purpose. Starting with a look at the relatively complex, but fairly powerful commands which produce colour, we will progress through drawing static pictures to producing animation on the screen.

Commands

To begin with, here is a summary of the relevant commands:

```
MODE n
BORDER c
INK p,c        or   INK p,c,d
PEN p          or   PEN#n,p
PAPER p        or   PAPER#n,p
LOCATE x,y     or   LOCATE#n,x,y
CHR$(n)
SYMBOL AFTER n
SYMBOL n, definition
PLOT x,y,i
PLOTR x,y,i
DRAW x,y,i
DRAWR x,y,i
MOVE x,y
MOVER x,y
TAG
TAGOFF
WINDOW#n,l,r,t,b
```

MODE n
This is the command which is used to set the screen mode and should be one of the first commands in any program. There are three screen modes (n=0,1,2):

```
MODE 2 : 80 characters across the screen     2 colours
MODE 1 : 40 characters across the screen     4 colours
MODE 0 : 20 characters across the screen    16 colours
```

Another effect of this command is to clear the screen to INK 0 (see below) and position the cursor in the top left-hand corner.

Although in MODE 2 you are permitted two colours, you must remember that you need a colour for the background and a colour for the characters, and therefore your two choices of colour are immediately consumed.

BORDER c

Around the usable screen is a border which can be set to any colour independent of the mode you are using. The BORDER command is followed by one or two integers in the range 0-26, specifying the colour of the border or two colours which the border will alternate between. The available colours and their numerical codes are displayed in the table below.

Ink number, c	Colour	Ink number,c	Colour
0	black	1	blue
2	bright blue	3	red
4	magenta	5	mauve
6	bright red	7	purple
8	bright magenta	8	green
10	cyan	11	sky blue
12	yellow	13	white
14	pastel blue	15	orange
16	pink	17	pastel magenta
18	bright green	19	sea green
20	bright cyan	21	lime green
22	pastel green	23	pastel cyan
24	bright yellow	25	pastel yellow
26	bright white		

To demonstrate these different colours the following program will change the colour of the border and print its corresponding number on the screen each time you press a key.

```
10 MODE 0
20 FOR I=0 TO 26
30 BORDER I
40 LOCATE 6,12:PRINT "COLOUR ";I
50 IF INKEY$="" THEN GOTO 50
60 CLS
70 NEXT I
```

```
80 LOCATE 7,12:PRINT "FINISHED"
90 FOR Z=1 TO 5000:NEXT Z
100 MODE 1
```

INK, PEN and PAPER

As mentioned above, in the different screen modes you are allowed either two, four or sixteen different colours and you have control over which colours the Amstrad will use. Let us consider MODE 1 as an example, since it is reasonably flexible in its use of colours, and see how different colours can be selected.

To instruct the machine in the colours of your choice, you must use the INK command, and this can be regarded as choosing a bottle of ink from the twenty-seven possible bottles. In MODE 1 you are permitted four possible colours and we will think of this as having four boxes (labelled box 0–box 3), into which the ink bottles can be placed.

If in box 0 you wish to place 'bright cyan', then you must first find the numerical code representing the colour from the above table. The table shows that 'bright cyan' is colour 20 and so you would issue the command:

```
INK 0,20
```

Note that the background will change to this colour.

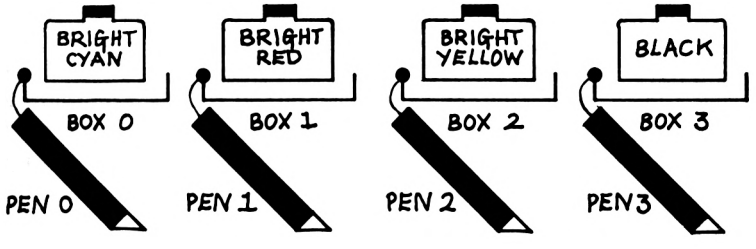
To select 'bright red' (colour 6) for box 1, the command would be

```
INK 1,6
```

Note that the characters will now be bright red.

Similarly, INK 2,24 places a bottle of 'bright yellow' ink into box 2, and INK 3,0 places 'black' ink into box 3.

After these four commands have been issued the situation is as shown below.



Now we are in a position to start writing in the four colours selected above, and from now on we refer to the box number.

To set the colour of the background or paper on which we are going to write, the PAPER command is used. For a black background, type

```
PAPER 3 (since black is in box 3)
```

and now the text is printed on a black background. To make the entire screen black, issue a CLS command.

Similarly, the combination

```
PAPER 2: CLS
```

will create a bright yellow screen.

You will notice that the PAPER command has no effect on the colour of the text (it is still red) and therefore the command PAPER 1 will result in the printing of red characters on a red background. Obviously the text cannot be seen! Restore normality by entering PAPER 2 (you will have to type this without being able to see it on the screen), followed by CLS.

To change the colour of the text we write with a different pen, since each box has a pen attached to it.

The command PEN 3 will change the colour of text to be printed into that of the ink found in box 3 (i.e. black).

Similarly PEN 0 will result in future text being printed in bright cyan (the colour of ink in box 0). It is possible to fit two bottles of ink into one box, and when this is done, the effect is to produce an output which will alternate in colour between the two selected. As an example of this, we will re-assign box 3 to contain black and white:

```
INK 3,0,26
```

Using the command PEN 3 will now result in text alternating in colour between black and white.

Programmers familiar with other machines may have encountered a RECOLOUR command, in which all the colours are re-defined. A similar effect can be produced on the Amstrad by re-defining an ink, i.e. placing a different bottle of ink into a previously used box. As an example of this, try the following program (type PEN 1: CLS to remove the flashing black and white characters):

```
10 MODE 1
20 INK 0,20:INK 1,6:INK 2,24:INK 3,0
30 CLS
40 PEN 1
50 PRINT
60 PRINT " AN EXAMPLE OF HOW TO CHANGE THE COLOUR
 OF ALREADY EXISTING TEXT"
70 PEN 3
80 END
```

The message is printed in PEN 1, i.e. the contents of box 1 – red.

Now type INK 1,2. The message should now appear in blue since we have placed colour 2 into box 1.

Try INK 1,15. The message is now orange since the numerical code for orange is 15.

If you type INK 1,20 you will have placed bright cyan in box 1. Although the message still exists it has been written in the same colour as the background, thus rendering it invisible.

LOCATE x,y
This command is used to place the text cursor anywhere on the screen, by giving the (x,y) co-ordinates of the position at which you wish to print. For example,

```
LOCATE 10,5:PRINT "HELLO"
```

will result in HELLO being displayed in the tenth column and the fifth row. Since the number of columns varies in the different screen modes, care must be taken to avoid giving invalid arguments, resulting in an error.

The following short program will introduce you to the different positions by experimenting with values for x and y.

```
10 INK 0,0:INK 1,26
20 MODE 1
30 LOCATE 2,2:INPUT"WHICH MODE (0-2) ";M
40 IF M<0 OR M>2 THEN GOTO 20
50 MODE 1
60 CLS:LOCATE 2,2:INPUT"GIVE THE VALUES OF X,Y ";X
,Y
70 MODE M
80 IF Y<1 OR Y>25 THEN GOTO 50
90 IF M=2 AND X>80 THEN GOTO 50
100 IF M=1 AND X>40 THEN GOTO 50
```

```
110 IF M=0 AND X>20 THEN GOTO 50
120 IF X<1 THEN GOTO 50
130 LOCATE X,Y:PRINT "H";
140 FOR Z=1 TO 5000:NEXT Z
150 GOTO 20
```

(Maps of the screen positions are given in Appendix VI of the Amstrad manual.)

CHR$ and user-defined graphics

The CHR$(n) function will convert a number into its character equivalent using the Amstrad's internal character set. A full list of the character set is given in Appendix III of the Amstrad manual, and the following routine will show all of the available characters on the screen.

```
10 MODE 1
20 FOR I=32 TO 255
30 PRINT CHR$(I);
40 NEXT I
50 END
```

By using CHR$ and LOCATE together, any character can be printed at any character position on the screen. For example,

```
LOCATE 10,10:PRINT CHR$(248)
```

will print a man at position (10,10).

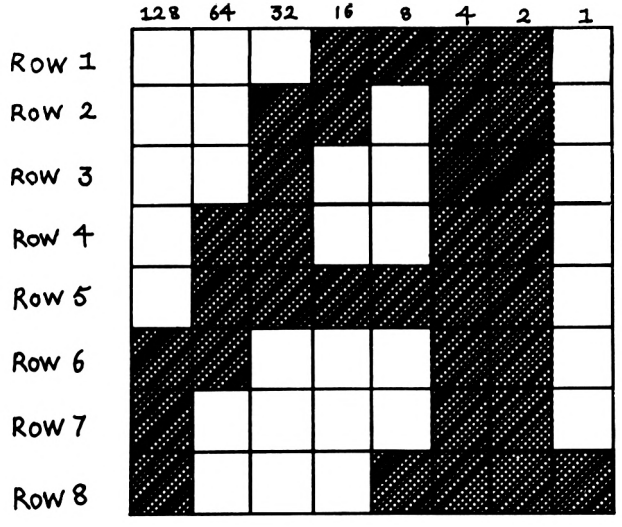
Although the Amstrad has a comprehensive character set and offers many useful characters, it also allows you to define your own. This is reasonably simple to do and is explained below.

First you must decide how many characters you wish to alter and issue a command informing the Amstrad at which point the characters will start to differ from its character set. This is done by issuing the command SYMBOL AFTER. For example,

```
SYMBOL AFTER 240
```

informs the computer that you will start to alter the character set at symbol 240.

Each character is formed by shading the squares on an eight by eight grid and then coding each row. For example, imagine that character 240 is to be re-defined as the letter A shown opposite.



The command to place this into the character set is

SYMBOL 240,row1,row2,row3,row4,row5,row6,row7,row8

where row1, row2 etc. are numerical codes for each of the rows. To construct the numerical codes, add up the numbers at the head of the columns which you wish to be shaded. For example,

row 1 16 + 8 + 4 + 2 =30
row 2 32 + 16 + 4 + 2 =54
row 3 32 + 4 + 2 =38 etc.

and the SYMBOL command becomes:

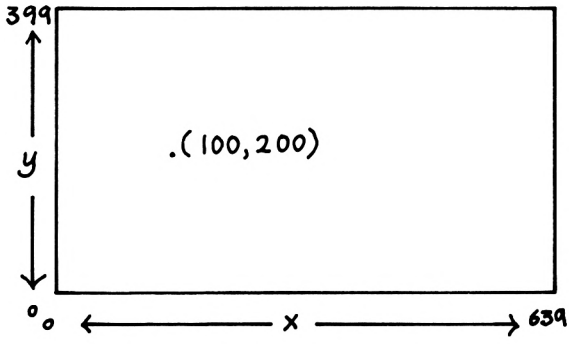
SYMBOL 240,30,54,38,102,126,198,134,143

This character can now be printed by using the command

PRINT CHR$(240)

PLOT, DRAW and MOVE

As mentioned in the section on the LOCATE command, there are screen maps for the three different modes of 20 × 25, 40 × 25 or 80 × 25. There is also a graphics map which divides the horizontal into 640 and the vertical into 400 individually plotable points, known as pixels.



The command PLOT 100, 200 will colour the pixel 100 across the screen and 200 up the screen (see diagram) and leave the graphics cursor at that position. To specify which colour it should be plotted in, a final argument is added to the command:

```
PLOT 100,200,2
```

which will plot the point using PEN 2 and will continue to use this pen in all future PLOT and DRAW commands. This does not, however, affect the pen which is being used to print characters.

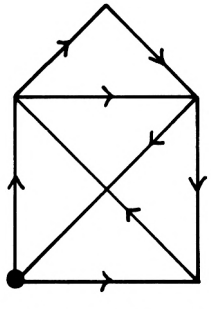
The DRAW x,y command will draw a straight line from the position of the graphics cursor to the position (x,y) given in the command. For example,

```
PLOT 100, 200:DRAW 300, 200
```

will draw a horizontal line from (100,200) to (300,200).

The following program shows how the DRAW command can be used:

```
10 MODE 1
20 PLOT 100,100,1
30 DRAW 100,200
40 DRAW 150;250
50 DRAW 200,200
60 DRAW 200,100
70 DRAW 100,200
80 DRAW 200,200
90 DRAW 100,100
100 DRAW 200,100
110 END
```



This will draw the shape without removing the pen from the paper.

Sometimes it is necessary to move to a location without drawing a line or plotting that point. This can be achieved by using the MOVE x,y command. For example,

```
MOVE 100,200
```

will move to the point (100,200) but not plot it.

PLOTR, DRAWR and MOVER
The effect of these commands is the same as PLOT, DRAW and MOVE, but the arguments contain slightly different information. The R in each of the commands stand for 'relative', i.e. in the command

```
DRAWR x,y
```

the movement (x,y) is relative to the current position of the graphics cursor.

The PLOT 100,100 command will move the graphic cursor to the point (100,100) and colour the pixel. If the command DRAWR 200,100 is now given the Amstrad will draw a line which moves 200 to the right and 100 up from the point (100,100). Consequently, the combination

```
PLOT 100,100:DRAW 300,200
```

produces the same effect as

```
PLOT 100,100:DRAWR 200,100
```

The following program will produce the same shape as was produced by the previous program by exactly the same route, but using the DRAWR command.

```
10 MODE 1
20 PLOT 100,100,1
30 DRAWR 0,100
40 DRAWR 50,50
50 DRAWR 50,-50
60 DRAWR 0,-100
70 DRAWR -100,100
80 DRAWR 100,0
90 DRAWR -100,-100
100 DRAWR 100,0
110 END
```

Notice that in line 50 the argument in the y-position is negative and that in line 70 the argument in the x-position is negative. These correspond to movement down the screen and across the screen to the left respectively.

It should be noted that although the range of values of x is 0-639 and the range for y is 0-399, points outside this range can be referred to, but exist off the visible screen. For example,

```
PLOT 100,100:DRAW 1000,100:DRAW 100,200
```

is quite legal, but the point (1000,100) is not visible.

TAG and TAGOFF

The TAG command is one of the most powerful graphics commands that Locomotive Basic offers because it allows you to print characters at the location of the graphics cursor, i.e. anywhere on the screen. However, there are some rules which you must pay heed to before you start to use the TAG command.

First, all PRINT statements must finish with a semi-colon (;), otherwise the machine will print the characters which it normally interprets as the return key.

Secondly, if animation is required, the character or group of characters should have a clear border all around the edge so as not to leave a trail when movement is in operation (see character 233 and Chapter 3).

The final and most important fact is that when TAG is employed, all characters will be printed with PAPER 0 as a background. So PAPER 0 acts as a 'global paper' since the background for all graphics using TAG will be that colour, although INK 0 can be defined as any of the 27 alternatives. If a different colour background is needed on the screen then you must employ a window which can have its own PAPER and PEN colours (see below).

The following program shows how animation can be created by using the TAG and MOVE commands:

```
10 MODE 1
20 BORDER 18
30 INK 0,11:INK 1,6
40 PLOT 1000,1000,1
50 FOR I=1 TO 100
60 TAG
70 X=20+I*2:Y=20
80 GOSUB 150
90 NEXT I
100 FOR J=1 TO 203
```

```
110 X=220+J*2:Y=20+(J^1.2/2)
120 GOSUB 150
130 NEXT J
140 END
150 REM ** JET **
160 MOVE X,Y
170 PRINT CHR$(128);CHR$(143);CHR$(143);CHR$(143);
CHR$(143);CHR$(143);CHR$(143);CHR$(246);
180 MOVE X,Y+16
190 PRINT CHR$(128);CHR$(215);
200 IF I=101 THEN GOTO 260
210 MOVE X+32,Y-16
220 PRINT CHR$(130);
230 MOVE X+80,Y-16
240 PRINT CHR$(130);
250 RETURN
260 MOVE X-16,Y-16
270 PRINT "        ";
280 RETURN
```

Comments

Lines 10-30	Initialise screen
Line 40	Plot point off screen to select PEN 2
Lines 50-90	Move plane along ground
Lines 100-140	Plane takes off
Lines 150-280	Draw plane

The global colour effect can be observed by adding the line

```
25 INK 2,26:PAPER 2: CLS
```

which sets the background to white. However, when the plane is drawn it continues to use PAPER 0, which is blue. The TAGOFF command will re-direct the text to the text cursor which is positioned independently of the graphics cursor (see LOCATE).

WINDOW

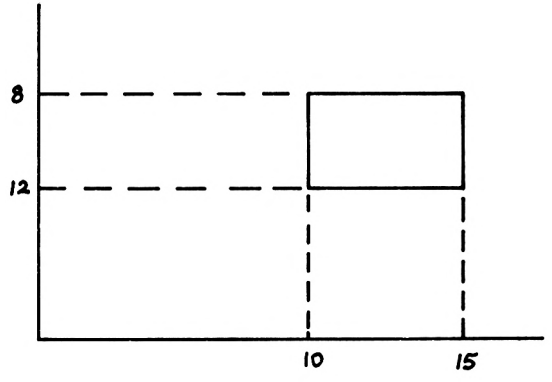
A window is an area of the screen which can be used to print textual messages while the remainder of the screen is dedicated to graphics. A window is defined by giving it a number n and the positions of its four sides:

```
WINDOW#n,l,r,t,b
```

where l is the left extremity, r the right extremity, t the top and b the bottom of the window n. For example,

WINDOW#1,10,15,8,12

will produce a window like the one show below.



Each window can have its own PEN and PAPER by setting the colours in that window. For example,

PAPER#1,3 PEN#1,2

Also the LOCATE command can be used in a window in order to position the text cursor. For example,

LOCATE#1,2,3:PRINT#1,"HI"

will select the 2nd column and 3rd row of the window.

Conclusion

To conclude, here is a program which contains most of the commands encountered in this section. We hope that it will provide you with a sound basis for producing your own graphics and animation programs.

```
10 INK 0,26
20 INK 1,9
30 INK 2,13
40 INK 3,2
50 PEN 1
60 BORDER 11
70 CLS
```

```
80 WINDOW#2,4,37,3,5
90 PAPER #2,3
100 PEN#2,0
110 CLS#2
120 SYMBOL AFTER 243
130 SYMBOL 246,0,0,0,0,0,0,1,1
140 SYMBOL 247,0,0,0,0,0,128,192,176
150 SYMBOL 248,4,15,31,63,254,248,127
160 SYMBOL 249,96,240,224,192,64,32,28,0
170 SYMBOL 250,0,0,0,0,0,1,3,7
180 SYMBOL 251,8,28,59,70,255,254,252,249
190 SYMBOL 252,7,7,15,30,62,54,70,70
200 SYMBOL 253,250,196,0,0,0,0,0,0
210 SYMBOL 254,0,0,1,1,3,6,2,0
220 SYMBOL 255,140,144,16,32,32,48,32,0
230 PAPER 0
240 PLOT 1000,1000,1
250 LET INC =5
260 MOVE 22,45:TAG
270 PRINT CHR$(246);CHR$(247);
280 MOVE 22,29:TAG
290 PRINT CHR$(248);CHR$(249);
300 PLOT 64,0,3:DRAW 559,0:PLOT 64,1:DRAW 559,1
310 PLOT 64,2:DRAW 559,2:PLOT 64,3:DRAW 559,3
320 TAGOFF:PEN 2
330 LOCATE 1,25:PRINT CHR$(143);CHR$(143);CHR$(143
);CHR$(143)
340 LOCATE 36,25:PRINT CHR$(143);CHR$(143);CHR$(14
3);CHR$(143)
350 FOR DEL=1 TO 800:NEXT
360 LET Y=76
370 LET M$="DE FROG IS JUMPING"
380 GOSUB 780
390 FOR X=30 TO 578 STEP 3
400 PLOT 0,0,3
410 PLOT 64,2:DRAW 559,2:PLOT 64,3:DRAW 559,3
420 PLOT 64,0:DRAW 559,0:PLOT 64,1:DRAW 559,1
430 TAGOFF:LOCATE 1,25:PRINT CHR$(143);CHR$(143);C
HR$(143);CHR$(143):LOCATE 36,25:PRINT CHR$(143);CH
R$(143);CHR$(143);CHR$(143)
440 PLOT 1000,1000,1
450 MOVE X-16,Y+16:TAG:PRINT"    ";
460 MOVE X-16,Y-16:TAG:PRINT" ";
470 IF INC>0 THEN MOVE X-16,Y-47:TAG:PRINT"    ";
480 MOVE X,Y:TAG
490 PRINT CHR$(250);CHR$(251);
500 MOVE X,Y-16:TAG
510 PRINT CHR$(252);CHR$(253);
520 MOVE X-16,Y-32
530 PRINT CHR$(254);CHR$(255);
```

```
540 LET Y=Y+INT(INC)
550 LET INC=INC-0.0503
560 IF INC<0 AND TI=0 THEN LET TI=1:LET M$="DE FRO
G IS GOING DOWN":GOSUB 780
570 NEXT X
580 MOVE X-16,Y+16:TAG:PRINT"   ";
590 MOVE X-16,Y-16:TAG:PRINT"    ";
600 MOVE X-16,Y-47:TAG:PRINT"   ";
610 MOVE X,Y:TAG
620 PRINT "  ";
630 MOVE X-16,Y-32
640 PRINT"  ";
650 MOVE X,Y-16:TAG
660 PRINT"  ";
670 MOVE X,Y-16:TAG
680 PRINT CHR$(246);CHR$(247);
690 MOVE X,Y-32:TAG
700 PRINT CHR$(248);CHR$(249);
710 TAGOFF:PEN 2:LOCATE 36,25:PRINT CHR$(143);CHR$
(143);CHR$(143);CHR$(143)
720 LET M$="DE FROG HAS LANDED"
730 GOSUB 780
740 FOR I=1 TO 20:LET Q$=INKEY$:NEXT I
750 IF INKEY$="" THEN GOTO 750
760 CLS:END
770 GOTO 770
780 CLS#2:LOCATE#2,INT((34-LEN(M$))/2)+1,2
790 PRINT#2,M$
800 RETURN
```

Comments

Lines 10-230	Initialise colours and user-defined graphics
Lines 240-390	Draw water, banks and frog
Lines 400-580	Move frog
Lines 590-740	Land frog
Lines 750-770	End routine
Lines 780-800	Print messages

Part One

Puzzles

3. Sliding Puzzle

This is a colourful animated version of the famous puzzle which requires you to recreate a picture on a four by four grid after the Amstrad has jumbled the pieces. There are fifteen pieces forming the picture and one empty position into which you can slide one of the adjacent pieces, thus moving them around the grid.

The computer will display the number of moves you have taken and when the original pattern is reproduced the puzzle is complete.

Playing instructions

When the program is executed the correct solution is displayed in the grid, together with two messages and a pattern explaining the letters corresponding to each of the grid positions. The messages are displayed while the Amstrad is constructing the puzzle, which will take about thirty seconds.

For reference, the grid pattern is as follows:

A	B	C	D
E	F	G	H
I	J	K	L
M	N	O	P

The empty position on the grid is shown by the white square, and this is the place which you can occupy with an adjacent piece. For example, if F is the empty position, you can slide either the tile at B, E, G, or J into it, but none other. To perform this operation, type the letter of the grid position you wish to vacate: e.g. if you decide

to move from grid position B in the above example, then you would press the B key.

If you exceed 999 moves the puzzle will end!

The program

One common feature of most puzzle programs is the necessity of a test routine to decide on the legality of a move. This is a display of artificial intelligence, since through this routine the computer 'knows' if you are trying to cheat. Clearly, if the puzzle is to be worth the RAM it occupies, this should not be allowed. It is important, therefore, that you take extra care when you enter the lines 720 to 1220, because an error in this information will result in the computer allowing illegal moves.

Flowchart

The flowchart (opposite) represents the general operation of the program.

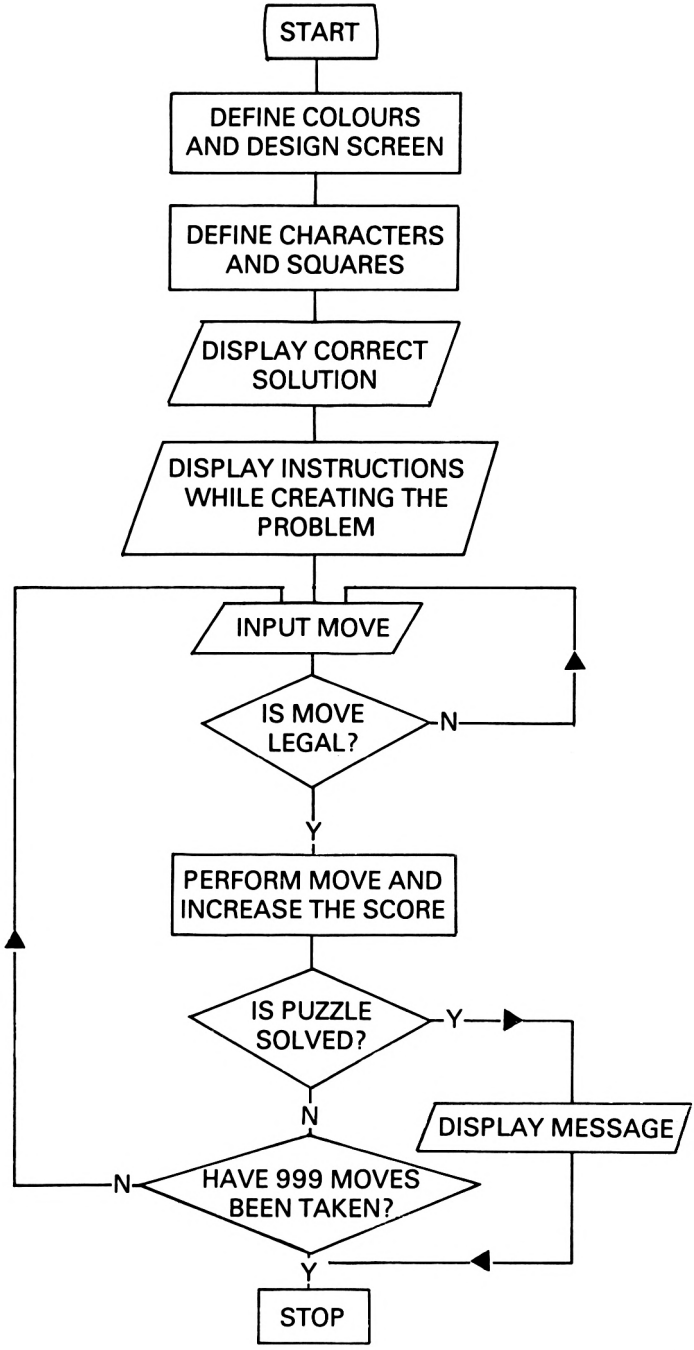
Variables

The following table represents the major variables used in the program and should help you to understand how the program operates.

P(x,y)	Array containing the characters of the picture
C(x,y)	Array for the colours of the characters
B()	Contains the position of the pieces
G	The number of moves
A	Direction of horizontal movement
U	Direction of vertical movement
BLANK	The piece which represents the empty square
SPACE	The board location of the empty square
B$	The letter of your move
B	The corresponding number to B$

Comments on the program

The program makes full use of subroutines, and each routine is clearly marked in the listing.


START
DEFINE COLOURS AND DESIGN SCREEN
DEFINE CHARACTERS AND SQUARES
DISPLAY CORRECT SOLUTION
DISPLAY INSTRUCTIONS WHILE CREATING THE PROBLEM
INPUT MOVE
IS MOVE LEGAL?
N
Y
PERFORM MOVE AND INCREASE THE SCORE
IS PUZZLE SOLVED?
Y
N
DISPLAY MESSAGE
HAVE 999 MOVES BEEN TAKEN?
N
Y
STOP

Lines 10-50
The opening section of the main program defines the colours to be used and initialises the screen display.

Lines 60-120
The remainder of the main program is concerned with calling the subroutines and displaying the correct solution in line 70.

Lines 130-240
This draws the grid.

Lines 250-290
The grid is outlined in this section.

Lines 300-500
The first of the move routines is to slide in the horizontal direction. First the position of the piece is computed and then by a sequence of MOVE, PLOT, TAG and PRINT commands, the piece is shifted either to the right or to the left (see Chapter 2). The newly formed empty grid position is then coloured white and the board is redrawn.

Lines 510-710
This is similar to the above lines but the resulting motion is vertical.

Lines 720-1220
This is the test routine to decide if a move is legal. If the move is accepted then this routine returns values of A and U which will represent the direction of the slide.

A=1 RIGHT	A=-1 LEFT
U=1 UP	U=-1 DOWN

Lines 1230-1350
This defines any characters which are needed (see below).

Lines 1360-1620
The pieces are defined (see below).

Lines 1630-1780
This draws a piece by computing the (x,y) co-ordinates of its top left-hand corner and then directing the textual characters to the graphics cursor with the TAG command.

Lines 1790-1900
This prints the instructions using a window to enable a different colour background and a different colour pen. The pen is defined in line 1840 and then both the pen and the paper are re-defined in line 1870 to use different colours for the display of the grid representation.

Lines 1910-1960
Another window is used to produce a title in different colours.

Lines 1970-2160
This is the routine which asks for your move. It allows you 999 moves, after which the program will end.

After the tests on legality have been completed, line 2120 swaps the pieces around in the computer's memory. If we need to swap the values of two variables, say A and B, we cannot simply type LET A=B: LET B=A, since this will result in both variables containing the value of B. This is because, by the time that we arrive at the second command (LET B=A), A already holds the value of B. To overcome this problem we must use a temporary store, e.g. Z, and issue the commands: LET Z=A: LET A=B: LET B=Z.

Lines 2170-2260
This routine informs the user that he has correctly solved the puzzle. It uses another window and PEN 2 which is defined in line 20 by the command INK 2,6,26. The two arguments mean that the pen will alternate in colour between red and white, and since there is a white background, this will appear on the screen as flashing red.

Lines 2270-2400
The computer plays the game for fifty moves to jumble the pieces.

Lines 2410-2490
This is another subroutine providing the user with further instructions.

Designing a new picture

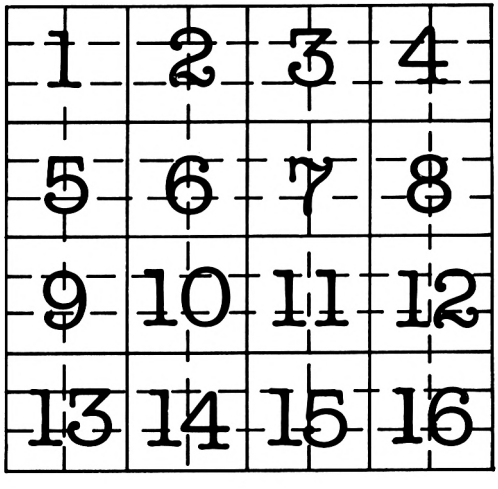
This is quite a difficult process, but with a little practice and experimentation it should be possible to construct pictures or patterns of your own design to use in the sliding puzzle program. If you consider lines 1460–1610 you will see sixteen data lines,

each line containing twelve numbers which represent the information for one of the pieces. But before you can create this coding for your own design we must examine how to form a picture.

As you will recall from Chapter 2, in MODE 0 you are permitted sixteen colours. In this program the first ten are defined in line 20, as these are the colours employed in the original design, but there is no reason why you should not supplement these by defining the remaining six. The defined colours are as follows:

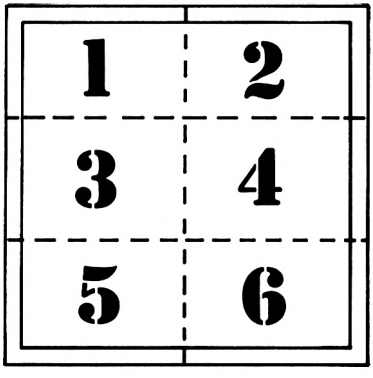
Number	Colour
0	bright cyan (global paper)
1	bright red
2	bright red/white (alternating)
3	magenta
4	bright green
5	bright yellow
6	bright blue
7	bright white
8	black
9	bright magenta

The picture is designed on a four by four grid with the pieces ordered as below:



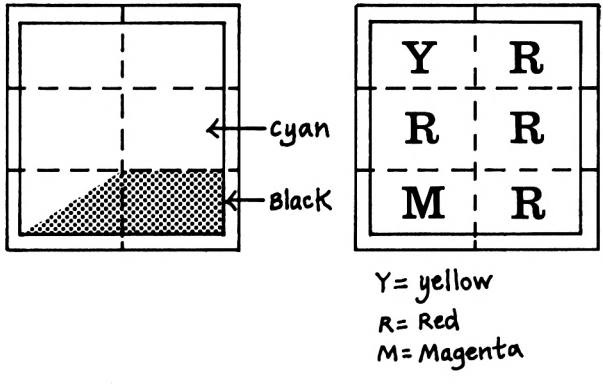
When designing a picture you should ensure that there is one blank square, and it is better if there are not two identical squares. There is no reason why you should not have two identical pieces, but, although they are identical to look at, to the computer they are different and must be returned to their own place of origin if the puzzle is to be completed correctly.

Each of the squares contained in the grid is formed by using six characters:



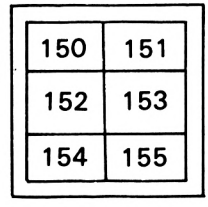
As mentioned in Chapter 2, when animation is required there must be a border of at least one pixel around the shape. If you wish to use colour, then each of the six characters can be coloured individually, but if you wish to design your own characters then you must take extra care. Because the TAG command always uses the global paper (INK 0), any character you produce will have a bright cyan background and this will normally restrict your design of picture.

These two different techniques can be seen by considering two lines from the picture in the program: lines 1470 and 1550.



In the left-hand square shown above, the bottom left-hand character will automatically have a global paper background and this restricts the colours for the remainder of the sky to the global colour. In the right-hand square the design is produced solely from blocks and these can be of any colour.

The program includes the six characters which fill the square completely except for the pixel border. The numerical values of these characters are given as



Any characters that you wish to design should therefore be positioned from 156 onwards and defined as shown in Chapter 2, placing them in the program somewhere between lines 1240 and 1350.

Note: do not forget the pixel border.

When your picture is complete it should be represented by a sequence of sixteen data lines stored from 1460 to 1610, with each line containing the numerical code for the six characters and their colours:

```
1500 DATA    Ch.1,Col.1,Ch.2,Col.2,Ch.3,Col.3,Ch.4,
             Col.4,Ch.5,Col.5,Ch.6,Col.6
```

where Ch.1 represents the numerical code for character 1 and Col.1 represents the corresponding colour. These lines should be in the order indicated in the grid on p.52 above.

The blank square should be entered as:

```
DATA 0,150,7,151,7,152,7,153,7,154,7,155,7
```

with the extra zero in the data informing the computer that this is the blank square.

The listing

The listing should be entered very carefully and then saved on to cassette as outlined in the Introduction.

The game should be entered and played in the caps lock mode. All spaces within the text should be entered as they appear.

```
10 MODE 0
20 INK 0,20:INK 1,6:INK 2,6,26:INK 3,4:INK 4,18:IN
K 5,24:INK 6,2:INK 7,26:INK 8,0:INK 9,8
30 WINDOW #1,2,8,7,21:PAPER #1,7:PEN #1,1:WINDOW #
4,10,18,21,21:PAPER #4,0:PEN #4,1
40 PAPER 5:PEN 1:CLS
50 BORDER 4
60 GOSUB 1910:GOSUB 1230:GOSUB 1360
70 FOR P=1 TO 16:B(P)=P:R=P:GOSUB 1630:NEXT P
80 GOSUB 130
90 GOSUB 1790
100 GOSUB 2270
110 GOSUB 1970
120 END
130 REM ** DRAW BOARD **
140 GOSUB 250
150 PLOT 300,300,6
160 DRAW 300,110:DRAW 555,110:DRAW 555,300:DRAW 30
0,300
170 DRAW 300,254:DRAW 555,254:DRAW 555,252:DRAW 30
0,252
180 DRAW 300,206:DRAW 555,206:DRAW 555,204:DRAW 30
0,204
190 DRAW 300,158:DRAW 555,158:DRAW 555,156:DRAW 30
0,156
200 MOVE 363,300:DRAW 363,110:DRAW 364,110:DRAW 36
4,300
210 MOVE 427,300:DRAW 427,110:DRAW 428,110:DRAW 42
8,300
220 MOVE 491,300:DRAW 491,110:DRAW 492,110:DRAW 49
2,300
230 PLOT 0,0,1
240 RETURN
250 REM ** DRAW BOARD OUTLINE **
260 PLOT 299,302,6
270 DRAW 299,108:DRAW 556,108:DRAW 556,302:DRAW 29
9,302
280 PLOT 0,0,1
290 RETURN
300 REM ** MOVE ACROSS **
310 LET X=236+64*(B-4*INT((B-1)/4))
```

```
320 LET R=B(B)
330 LET Y=300-48*INT((B-1)/4)
340 FOR N=0 TO 64*A STEP A
350 PLOT X+N,Y,C(R,1)
360 TAG:PRINT CHR$(P(R,1));:TAGOFF
370 PLOT X+32+N,Y,C(R,2)
380 TAG:PRINT CHR$(P(R,2));:TAGOFF
390 PLOT X+N,Y-16,C(R,3)
400 TAG:PRINT CHR$(P(R,3));:TAGOFF
410 PLOT X+32+N,Y-16,C(R,4)
420 TAG:PRINT CHR$(P(R,4));:TAGOFF
430 PLOT X+N,Y-32,C(R,5)
440 TAG:PRINT CHR$(P(R,5));:TAGOFF
450 PLOT X+32+N,Y-32,C(R,6)
460 TAG:PRINT CHR$(P(R,6));:TAGOFF
470 NEXT N
480 LET P=B:LET R=BLANK:GOSUB 1630
490 GOSUB 130
500 RETURN
510 REM ** MOVE UP/DOWN **
520 LET X=236+64*(B-4*INT((B-1)/4))
530 LET R=B(B)
540 LET Y=300-48*INT((B-1)/4)
550 FOR N=0 TO U*48 STEP U
560 PLOT X,Y+N,C(R,1)
570 TAG:PRINT CHR$(P(R,1));:TAGOFF
580 PLOT X+32,Y+N,C(R,2)
590 TAG:PRINT CHR$(P(R,2));:TAGOFF
600 PLOT X,Y-16+N,C(R,3)
610 TAG:PRINT CHR$(P(R,3));:TAGOFF
620 PLOT X+32,Y-16+N,C(R,4)
630 TAG:PRINT CHR$(P(R,4));:TAGOFF
640 PLOT X,Y-32+N,C(R,5)
650 TAG:PRINT CHR$(P(R,5));:TAGOFF
660 PLOT X+32,Y-32+N,C(R,6)
670 TAG:PRINT CHR$(P(R,6));:TAGOFF
680 NEXT N
690 LET P=B:LET R=BLANK:GOSUB 1630
700 GOSUB 130
710 RETURN
720 REM ** TEST ROUTINE **
730 LET A=0:LET U=0
740 IF B=1 AND SPACE=2 THEN LET A=1:RETURN
750 IF B=1 AND SPACE=5 THEN LET U=-1:RETURN
760 IF B=2 AND SPACE=1 THEN LET A=-1:RETURN
770 IF B=2 AND SPACE=3 THEN LET A=1:RETURN
780 IF B=2 AND SPACE=6 THEN LET U=-1:RETURN
790 IF B=3 AND SPACE=2 THEN LET A=-1:RETURN
800 IF B=3 AND SPACE=4 THEN LET A=1:RETURN
810 IF B=3 AND SPACE=7 THEN LET U=-1:RETURN
820 IF B=4 AND SPACE=3 THEN LET A=-1:RETURN
```

```
830 IF B=4 AND SPACE=8 THEN LET U=-1:RETURN
840 IF B=5 AND SPACE=1 THEN LET U=1:RETURN
850 IF B=5 AND SPACE=6 THEN LET A=1:RETURN
860 IF B=5 AND SPACE=9 THEN LET U=-1:RETURN
870 IF B=6 AND SPACE=5 THEN LET A=-1:RETURN
880 IF B=6 AND SPACE=7 THEN LET A=1:RETURN
890 IF B=6 AND SPACE=2 THEN LET U=1:RETURN
900 IF B=6 AND SPACE=10 THEN LET U=-1:RETURN
910 IF B=7 AND SPACE=6 THEN LET A=-1:RETURN
920 IF B=7 AND SPACE=8 THEN LET A=1:RETURN
930 IF B=7 AND SPACE=3 THEN LET U=1:RETURN
940 IF B=7 AND SPACE=11 THEN LET U=-1:RETURN
950 IF B=8 AND SPACE=7 THEN LET A=-1:RETURN
960 IF B=8 AND SPACE=4 THEN LET U=1:RETURN
970 IF B=8 AND SPACE=12 THEN LET U=-1:RETURN
980 IF B=9 AND SPACE=5 THEN LET U=1:RETURN
990 IF B=9 AND SPACE=10 THEN LET A=1:RETURN
1000 IF B=9 AND SPACE=13 THEN LET U=-1:RETURN
1010 IF B=10 AND SPACE=9 THEN LET A=-1:RETURN
1020 IF B=10 AND SPACE=11 THEN LET A=1:RETURN
1030 IF B=10 AND SPACE=6 THEN LET U=1:RETURN
1040 IF B=10 AND SPACE=14 THEN LET U=-1:RETURN
1050 IF B=11 AND SPACE=10 THEN LET A=-1:RETURN
1060 IF B=11 AND SPACE=12 THEN LET A=1:RETURN
1070 IF B=11 AND SPACE=15 THEN LET U=-1:RETURN
1080 IF B=11 AND SPACE=7 THEN LET U=1:RETURN
1090 IF B=12 AND SPACE=11 THEN LET A=-1:RETURN
1100 IF B=12 AND SPACE=8 THEN LET U=1:RETURN
1110 IF B=12 AND SPACE=16 THEN LET U=-1:RETURN
1120 IF B=13 AND SPACE=9 THEN LET U=1:RETURN
1130 IF B=13 AND SPACE=14 THEN LET A=1:RETURN
1140 IF B=14 AND SPACE=13 THEN LET A=-1:RETURN
1150 IF B=14 AND SPACE=15 THEN LET A=1:RETURN
1160 IF B=14 AND SPACE=10 THEN LET U=1:RETURN
1170 IF B=15 AND SPACE=11 THEN LET U=1:RETURN
1180 IF B=15 AND SPACE=14 THEN LET A=-1:RETURN
1190 IF B=15 AND SPACE=16 THEN LET A=1:RETURN
1200 IF B=16 AND SPACE=15 THEN LET A=-1:RETURN
1210 IF B=16 AND SPACE=12 THEN LET U=1:RETURN
1220 RETURN
1230 REM ** DEFINE CHARS **
1240 SYMBOL AFTER 150
1250 SYMBOL 150,&0,&7F,&7F,&7F,&7F,&7F,&7F,&7F
1260 SYMBOL 151,&0,&FE,&FE,&FE,&FE,&FE,&FE,&FE
1270 SYMBOL 152,&7F,&7F,&7F,&7F,&7F,&7F,&7F,&7F
1280 SYMBOL 153,&FE,&FE,&FE,&FE,&FE,&FE,&FE,&FE
1290 SYMBOL 154,&7F,&7F,&7F,&7F,&7F,&7F,&7F,&0
1300 SYMBOL 155,&FE,&FE,&FE,&FE,&FE,&FE,&FE,&0
1310 SYMBOL 156,&0,&0,&0,&0,&0,&0,&0,&7F
1320 SYMBOL 157,&0,&2,&6,&E,&1E,&3E,&7E,&FE
1330 SYMBOL 158,&1,&3,&7,&F,&1F,&3F,&7F,&0
```

```
1340 SYMBOL 159,&80,&C0,&E0,&F0,&F8,&FC,&FE,&0
1350 RETURN
1360 REM ** DEFINE SQUARES **
1370 DIM P(16,6),C(16,6),B(16)
1380 FOR I=1 TO 16
1390 FOR J=1 TO 6
1400 READ P(I,J)
1410 IF P(I,1)=0 THEN LET BLANK=I:GOTO 1400
1420 READ C(I,J)
1430 NEXT J
1440 NEXT I
1450 LET SPACE=BLANK
1460 DATA 0,150,7,151,7,152,7,153,7,154,7,155,7
1470 DATA 128,0,128,0,128,0,128,0,158,8,155,8
1480 DATA 128,0,128,0,128,0,128,0,154,8,159,8
1490 DATA 128,0,128,0,128,0,128,0,154,8,128,8
1500 DATA 128,0,157,8,128,0,153,1,128,0,155,1
1510 DATA 150,8,151,8,152,1,153,1,154,5,155,1
1520 DATA 150,8,151,8,152,1,153,1,154,1,155,5
1530 DATA 150,8,128,0,152,1,128,0,154,1,128,0
1540 DATA 128,0,151,1,128,0,153,1,154,4,155,1
1550 DATA 150,5,151,1,152,1,153,1,154,9,155,1
1560 DATA 150,1,151,5,152,1,153,1,154,5,155,5
1570 DATA 150,1,128,0,152,1,128,0,154,1,155,4
1580 DATA 150,4,151,1,152,4,153,1,154,4,155,4
1590 DATA 150,9,151,1,152,9,153,1,154,4,155,4
1600 DATA 150,5,151,5,152,1,153,1,154,4,155,4
1610 DATA 150,1,151,4,152,1,153,4,154,4,155,4
1620 RETURN
1630 REM ** DRAW PIECES **
1640 LET X=236+64*(P-4*INT((P-1)/4))
1650 LET Y=300-48*INT((P-1)/4)
1660 PLOT X,Y,C(R,1)
1670 TAG:PRINT CHR$(P(R,1));:TAGOFF
1680 PLOT X+32,Y,C(R,2)
1690 TAG:PRINT CHR$(P(R,2));:TAGOFF
1700 PLOT X,Y-16,C(R,3)
1710 TAG:PRINT CHR$(P(R,3));:TAGOFF
1720 PLOT X+32,Y-16,C(R,4)
1730 TAG:PRINT CHR$(P(R,4));:TAGOFF
1740 PLOT X,Y-32,C(R,5)
1750 TAG:PRINT CHR$(P(R,5));:TAGOFF
1760 PLOT X+32,Y-32,C(R,6)
1770 TAG:PRINT CHR$(P(R,6));:TAGOFF
1780 RETURN
1790 REM ** INSTRUCTIONS **
1800 CLS #1
1810 PLOT 30,62,8:DRAW 30,304:DRAW 256,304:DRAW 25
6,62:DRAW 30,62
1820 PLOT 30,176,8:DRAW 256,176
1830 PRINT #1
```

```
1840 PEN #1,8
1850 PRINT #1,"  THE":PRINT #1,"GRID IS  AS":PRINT
 #1,"FOLLOWS"
1860 PRINT #1:PRINT #1
1870 PEN #1,6:PAPER #1,0
1880 PRINT #1,"A B C D       E F G H       I J K L
       M N O P";
1890 PAPER #1,7:PEN #1,1
1900 RETURN
1910 REM ** HEADING **
1920 WINDOW #2,3,18,2,4:PEN #2,1:PAPER #2,0
1930 CLS #2
1940 PEN 8:PLOT 62,334,8:DRAW 62,384:DRAW 576,384:
DRAW 576,334:DRAW 62,334
1950 PRINT #2:PRINT #2," SLIDING PUZZLE"
1960 RETURN
1970 REM ** INPUT MOVE **
1980 WINDOW #3,2,8,7,14:PAPER #3,7:PEN #3,1
1990 CLS #4:PRINT #4,"MOVES ";:PRINT #4, USING "##
#";G;
2000 PLOT 284,62,8:DRAW 284,80:DRAW 576,80:DRAW 57
6,62:DRAW 284,62
2010 CLS #3
2020 PRINT #3:PRINT #3:PRINT #3," ENTER":PRINT #3,
" PIECE":PRINT #3," TO BE":PRINT #3," MOVED"
2030 PLOT 30,176,8:DRAW 256,176
2040 FOR G=1 TO 999
2050 LET B$=INKEY$:IF B$="" THEN GOTO 2050
2060 IF B$<"A" OR B$>"P" THEN GOTO 2050
2070 LET B=ASC(B$)-64
2080 GOSUB 720
2090 IF U=0 AND A=0 THEN GOTO 2050
2100 IF U=0 THEN GOSUB 300
2110 IF A=0 THEN GOSUB 510
2120 LET S=SPACE:LET SPACE=B:LET Z=B(SPACE):LET B(
SPACE)=B(S):LET B(S)=Z
2130 PRINT #4,"MOVES ";:PRINT #4, USING "###";G;
2140 GOSUB 2170
2150 NEXT G
2160 RETURN
2170 REM ** SOLVED **
2180 FOR I=1 TO 16
2190 IF B(I)<>I THEN RETURN
2200 NEXT I
2210 WINDOW #5,5,5,9,19:PAPER #5,7:PEN #5,2
2220 CLS #1
2230 PRINT #5,"S O L V E D";
2240 IF INKEY$="" THEN GOTO 2240
2250 CLS
2260 END
2270 REM ** CREATE PROBLEM **
```

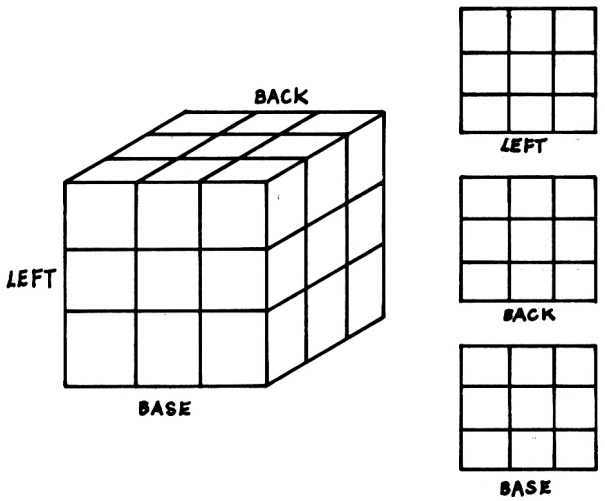
```
2280 FOR I=1 TO 50
2290 IF I=26 THEN GOSUB 2410
2300 LET B=1+INT(16*RND(1))
2310 GOSUB 720
2320 IF A=0 AND U=0 THEN GOTO 2300
2330 LET S=SPACE:LET SPACE=B:LET Z=B(SPACE):LET B(
SPACE)=B(S):LET B(S)=Z
2340 NEXT I
2350 FOR P=1 TO 16
2360 LET R=B(P)
2370 GOSUB 1630
2380 NEXT P
2390 GOSUB 130
2400 RETURN
2410 REM ** INSTRUCTIONS 2 **
2420 CLS #1
2430 PEN #1,8
2440 PRINT #1," ENTER  THE    SQUARE YOU    WISH
 TO     MOVE   FROM"
2450 PEN #1,6:PAPER #1,0
2460 PRINT #1,"A B C D       E F G H       I J K L
      M N O P";
2470 PAPER #1,7:PEN #1,1
2480 PLOT 30,176,8:DRAW 256,176
2490 RETURN
```

4. The Cube

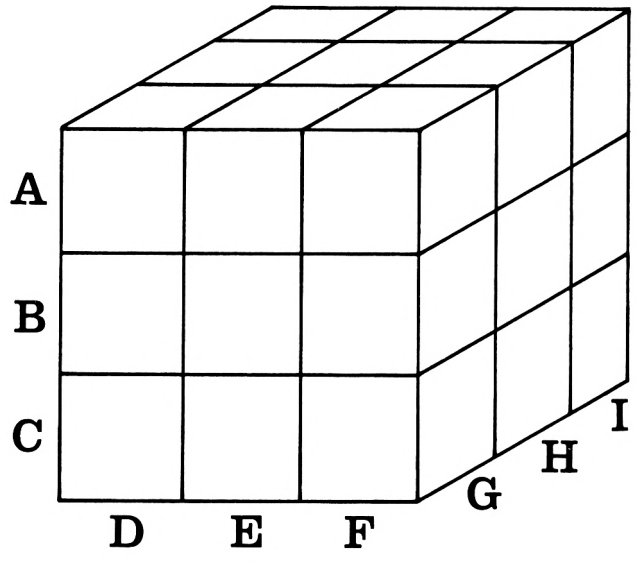
The cube is a three-dimensional graphical representation of the famous Rubik Cube, with the computer generating its picture and then mixing it up. Your task is to solve the puzzle by finishing with each of the six faces containing pieces of only one colour. When you have input your move, the colours on each face of the cube will change automatically according to your instruction.

Playing instructions

When the program is executed, the following will be displayed on the screen:



When you wish to move a section of the cube, type the letter referring to that section (see diagram on p.62) and then input the direction you wish to move by using one of the arrow keys: e.g. to



rotate the front face clockwise, you would enter 'G' followed by the down arrow. If you decide to enter a sequence of moves then type 'S' before entering the sequence and then 'P' to finish.

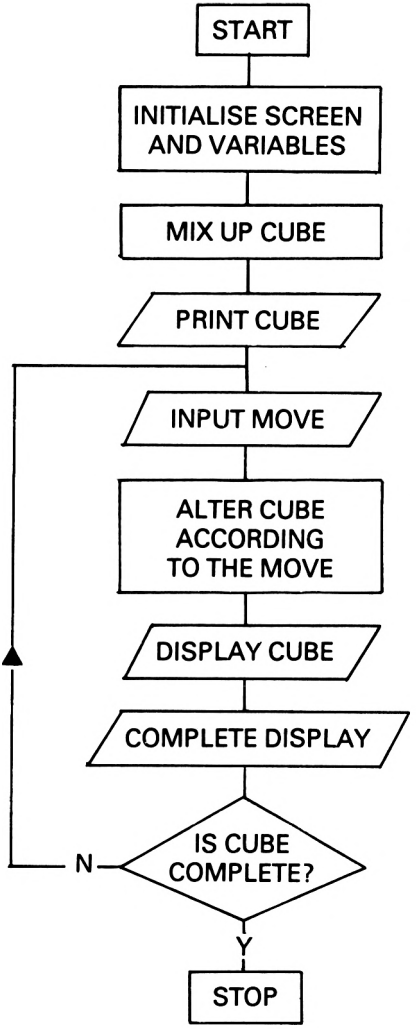
The game will end when each face has only one colour covering it.

The program

Due to the nature of this and other programs in this book, particular sections relating to the logistics of the game or puzzle are quite complex and should be entered with great care. In this program, lines 670–1780 are a prime example since this is the routine which performs the rotation which the user has requested. Any mistake in this code will cause dire confusion when trying to solve the puzzle.

Flowchart

The flowchart (opposite) represents the general operation of the program.



Variables

The following table represents the major variables used in the program and should help you to understand how the program operates.

F(x,y)	Front face colours
B(x,y)	Bottom face colours
L(x,y)	Left face colours
R(x,y)	Right face colours
T(x,y)	Top face colours
BA(x,y)	Back face colours
Z()	Arrays for the temporary storage of colours
Z1()	
Z2()	
KE$	The section to be moved
DIR	The direction the section is to be moved in
S	Sequence state

Comments on the program

The program is highly structured with the main body of the program calling the various subroutines. It is written to use MODE 0, since this is the only mode which provides sufficient colours to make the graphics practical.

Lines 10-150
This routine initialises the variables and arrays required for the program. Each face has its own two-dimensional array in which the colours of the squares are stored. Eight ink colours are used in this program and these are defined in line 140.

Line 160
The completed cube is displayed, giving the player a look at the ultimate goal of the puzzle.

Lines 170-240
This section of the program mixes the pieces of the cube by performing a sequence of ten legal moves after which the cube is re-displayed on the screen.

Lines 250-310
These statements form the main body of the program from which all the other routines are called. This section is repeated until the routine at line 320 detects that the puzzle has been completed.

Lines 320-440
This routine checks to see if the puzzle has been completed by comparing all the squares on a face with the centre square. If they all match, then the puzzle is finished.

Lines 450-660
In this section the player inputs his move, which is then checked to ensure it is legal. If legal, the move is highlighted on the screen before the direction is input.

Lines 670-1780
The player's move is made and the arrays which hold the colours for each of the faces are altered accordingly. Note the heavy use of FOR NEXT loops to make the program more efficient.

Lines 1790-2490
This routine displays the cube and the three hidden faces on the screen.

The listing

The listing should be entered very carefully and then saved on to cassette as outlined in the Introduction.

The game should be entered and played in the caps lock mode. All spaces within the text should be entered as they appear.

```
10 MODE 0
20 DIM Z1(3),Z(3),F(3,3),B(3,3),L(3,3),R(3,3),T(3,
3),BA(3,3)
30 FOR I=1 TO 3:FOR K=1 TO 3
40 RESTORE
50 READ A:F(I,K)=A
60 READ A:B(I,K)=A
70 READ A:L(I,K)=A
80 READ A:R(I,K)=A
90 READ A:T(I,K)=A
100 READ A:BA(I,K)=A
110 NEXT K,I
120 DATA 2,3,4,5,6,7
130 INK 8,25
140 INK 0,0:INK 1,24:INK 2,13:INK 3,1:INK 4,3:INK
5,19:INK 6,15:INK 7,25
150 BORDER 0
160 GOSUB 1790
170 PRINT"": LOCATE 1,6:PRINT"MIXING"
180 FOR I=1 TO 10
190 KE$=CHR$(65+INT(RND*6))
200 DIR =INT (RND*2)+1
210 GOSUB 670
220 NEXT I
230 LOCATE 1,6:PRINT"       "
```

```
240 GOSUB 1790
250 GOSUB 450
260 LOCATE 1,6:PRINT"            "
270 GOSUB 670
280 IF S=0 THEN GOSUB 1790
290 GOSUB 2400
300 GOSUB 320
310 GOTO 250
320 FOR XC=1 TO 3
330 FOR YC=1 TO 3
340 IF F(XC,YC)<>F(2,2)THEN RETURN
350 IF B(XC,YC)<>B(2,2)THEN RETURN
360 IF L(XC,YC)<>L(2,2)THEN RETURN
370 IF R(XC,YC)<>R(2,2)THEN RETURN
380 IF T(XC,YC)<>T(2,2)THEN RETURN
390 IF BA(XC,YC)<>BA(2,2)THEN RETURN
400 NEXT YC,XC
410 FOR I=1000 TO 16 STEP -1:SOUND 1,I,2:SOUND 2,I
*2,2:SOUND 4,I/2,2:INK INT(RND*8),INT (RND*25):NEX
T I
420 INK 0,0:INK 1,24
430 LOCATE 1,6:PRINT"PUZZLE":LOCATE 1,7:PRINT"COMP
LETE"
440 GOTO 440
450 LOCATE 1,6:PRINT CHR$(7)+"MOVE ?"
460 Q$=INKEY$:IF Q$=""THEN 460
470 IF Q$="A"THEN PEN 8:PLOT 70,175,8:TAG:PRINT"A"
;:TAGOFF:KE$="A":GOTO 590
480 IF Q$="B"THEN PEN 8:PLOT 70,135,8:TAG:PRINT"B"
;:TAGOFF:KE$="B":GOTO 590
490 IF Q$="C"THEN PEN 8:PLOT 70,95,8:TAG:PRINT"C";
:TAGOFF:KE$="C":GOTO 590
500 IF Q$="D"THEN PEN 8:PLOT 115,60,8:TAG:PRINT"D"
;:TAGOFF:KE$="D":GOTO 590
510 IF Q$="E"THEN PEN 8:PLOT 157,60,8:TAG:PRINT"E"
;:TAGOFF:KE$="E":GOTO 590
520 IF Q$="F"THEN PEN 8:PLOT 200,60,8:TAG:PRINT"F"
;:TAGOFF:KE$="F":GOTO 590
530 IF Q$="G"THEN PEN 8:PLOT 244,78,8:TAG:PRINT"G"
;:TAGOFF:KE$="G":GOTO 590
540 IF Q$="H"THEN PEN 8:PLOT 278,105,8:TAG:PRINT"H
";:TAGOFF:KE$="H":GOTO 590
550 IF Q$="I"THEN PEN 8:PLOT 310,128,8:TAG:PRINT"I
";:TAGOFF:KE$="I":GOTO 590
560 IF Q$="S"THEN S=1:GOTO 250
570 IF Q$="P"THEN S=0:LOCATE 1,6:PRINT"      ":GOS
UB 1790:GOTO 250
580 GOTO 460
590 PEN 1:PRINT"":LOCATE 1,6:PRINT"DIRECTION ?"
600 IF KE$>"C"THEN 640
610 IF INKEY(1)=0 THEN DIR=2:RETURN
```

```
620 IF INKEY(8)=0 THEN DIR=1:RETURN
630 GOTO 610
640 IF INKEY(0)=0 THEN DIR=2:RETURN
650 IF INKEY(2)=0 THEN DIR=1:RETURN
660 GOTO 640
670 IF KE$<>"A"THEN 790
680 FOR Q=1 TO 3:Z(Q)=F(Q,3):Z1(Q)=T(Q,1):NEXT Q
690 IF DIR<>1 THEN 740
700 FOR Q=1 TO 3
710 F(Q,3)=R(Q,3):R(Q,3)=B(Q,3):B(Q,3)=L(Q,3):L(Q,
3)=Z(Q):NEXT Q
720 T(1,1)=T(3,1):T(2,1)=T(3,2):T(3,1)=T(3,3):T(3,
2)=T(2,3):T(3,3)=T(1,3):T(2,3)=T(1,2)
730 T(1,3)=Z1(1):T(1,2)=Z1(2):RETURN
740 FOR Q=1 TO 3
750 F(Q,3)=L(Q,3):L(Q,3)=B(Q,3):B(Q,3)=R(Q,3):R(Q,
3)=Z(Q)
760 NEXT Q
770 T(2,1)=T(1,2):T(1,1)=T(1,3):T(1,2)=T(2,3):T(1,
3)=T(3,3):T(2,3)=T(3,2):T(3,3)=Z1(3):T(3,2)=Z1(2):
T(3,1)=Z1(1)
780 RETURN
790 IF KE$<>"B" THEN 900
800 FOR Q=1 TO 3:Z(Q)=F(Q,2):NEXT Q
810 IF DIR<>1 THEN 860
820 FOR Q=1 TO 3
830 F(Q,2)=R(Q,2):R(Q,2)=B(Q,2):B(Q,2)=L(Q,2):L(Q,
2)=Z(Q)
840 NEXT Q
850 RETURN
860 FOR Q=1 TO 3
870 F(Q,2)=L(Q,2):L(Q,2)=B(Q,2):B(Q,2)=R(Q,2):R(Q,
2)=Z(Q)
880 NEXT Q
890 RETURN
900 IF KE$<>"C" THEN 1030
910 FOR Q=1 TO 3:Z(Q)=F(Q,1):Z1(Q)=BA(Q,1):NEXT Q
920 IF DIR<>1 THEN 980
930 FOR Q=1 TO 3
940 F(Q,1)=R(Q,1):R(Q,1)=B(Q,1):B(Q,1)=L(Q,1):L(Q,
1)=Z(Q)
950 NEXT Q
960 BA(3,1)=BA(1,1):BA(2,1)=BA(1,2):BA(1,1)=BA(1,3
):BA(1,2)=BA(2,3):BA(1,3)=BA(3,3):BA(2,3)=BA(3,2):
BA(3,3)=Z1(3):BA(3,2)=Z1(2)
970 RETURN
980 FOR Q=1 TO 3
990 F(Q,1)=L(Q,1):L(Q,1)=B(Q,1):B(Q,1)=R(Q,1):R(Q,
1)=Z(Q)
1000 NEXT Q
1010 BA(1,1)=BA(3,1):BA(2,1)=BA(3,2):BA(3,1)=BA(3,
```

```
3):BA(3,2)=BA(2,3):BA(3,3)=BA(1,3):BA(2,3)=BA(1,2)
:BA(1,3)=Z1(1):BA(1,2)=Z1(2)
1020 RETURN
1030 IF KE$<>"D"THEN 1170
1040 FOR Q=1 TO 3:Z(Q)=F(1,Q):Z1(Q)=L(Q,1):NEXT Q
1050 IF DIR<>1 THEN 1110
1060 FOR Q=1 TO 3
1070 F(1,Q)=T(1,Q):T(1,Q)=B(3,4-Q):B(3,4-Q)=BA(1,Q
):BA(1,Q)=Z(Q)
1080 NEXT Q
1090 L(1,1)=L(3,1):L(2,1)=L(3,2):L(3,1)=L(3,3):L(3
,2)=L(2,3):L(3,3)=L(1,3):L(2,3)=L(1,2):L(1,3)=Z1(1
):L(1,2)=Z1(2)
1100 RETURN
1110 FOR Q=1 TO 3
1120 Z1(Q)=L(1,Q)
1130 F(1,Q)=BA(1,Q):BA(1,Q)=B(3,4-Q):B(3,4-Q)=T(1,
Q):T(1,Q)=Z(Q)
1140 NEXT Q
1150 L(1,1)=L(1,3):L(1,2)=L(2,3):L(1,3)=L(3,3):L(2
,3)=L(3,2):L(3,3)=L(3,1):L(3,2)=L(2,1):L(3,1)=Z1(1
):L(2,1)=Z1(2)
1160 RETURN
1170 IF KE$<>"F"THEN 1300
1180 FOR Q=1 TO 3:Z(Q)=F(3,Q):Z1(Q)=R(1,Q):Z2(Q)=R
(Q,1):NEXT Q
1190 IF DIR<>1 THEN 1250
1200 FOR Q=1 TO 3
1210 F(3,Q)=T(3,Q):T(3,Q)=B(1,4-Q):B(1,4-Q)=BA(3,Q
):BA(3,Q)=Z(Q)
1220 NEXT Q
1230 R(1,1)=R(1,3):R(1,2)=R(2,3):R(1,3)=R(3,3):R(2
,3)=R(3,2):R(3,3)=R(3,1):R(3,2)=R(2,1):R(3,1)=Z1(1
):R(2,1)=Z1(2)
1240 RETURN
1250 FOR Q=1 TO 3
1260 F(3,Q)=BA(3,Q):BA(3,Q)=B(1,4-Q):B(1,4-Q)=T(3,
Q):T(3,Q)=Z(Q)
1270 NEXT Q
1280 R(1,1)=R(3,1):R(2,1)=R(3,2):R(3,1)=R(3,3):R(3
,2)=R(2,3):R(3,3)=R(1,3):R(2,3)=R(1,2)
1290 R(1,3)=Z2(1):R(1,2)=Z2(2):RETURN
1300 IF KE$<>"E"THEN 1410
1310 FOR Q=1 TO 3:Z(Q)=F(2,Q):NEXT Q
1320 IF DIR<>1 THEN 1370
1330 FOR Q=1 TO 3
1340 F(2,Q)=T(2,Q):T(2,Q)=B(2,4-Q):B(2,4-Q)=BA(2,Q
):BA(2,Q)=Z(Q)
1350 NEXT Q
1360 RETURN
1370 FOR Q=1 TO 3
```

```
1380 F(2,Q)=BA(2,Q):BA(2,Q)=B(2,4-Q):B(2,4-Q)=T(2,
Q):T(2,Q)=Z(Q)
1390 NEXT Q
1400 RETURN
1410 IF KE$<>"G"THEN 1550
1420 FOR Q=1 TO 3:Z(Q)=R(1,Q):Z1(Q)=F(Q,1):NEXT Q
1430 IF DIR<>1 THEN 1490
1440 FOR Q=1 TO 3
1450 R(1,Q)=T(4-Q,1):T(4-Q,1)=L(3,4-Q):L(3,4-Q)=BA
(Q,3):BA(Q,3)=Z(Q)
1460 NEXT Q
1470 F(1,1)=F(3,1):F(2,1)=F(3,2):F(3,1)=F(3,3):F(3
,2)=F(2,3):F(3,3)=F(1,3):F(2,3)=F(1,2)
1480 F(1,3)=Z1(1):F(1,2)=Z1(2):RETURN
1490 FOR Q=1 TO 3
1500 Z1(Q)=F(1,Q)
1510 R(1,4-Q)=BA(4-Q,3):BA(4-Q,3)=L(3,Q):L(3,Q)=T(
Q,1):T(Q,1)=Z(4-Q)
1520 NEXT Q
1530 F(1,1)=F(1,3):F(1,2)=F(2,3):F(1,3)=F(3,3):F(2
,3)=F(3,2):F(3,3)=F(3,1):F(3,2)=F(2,1):F(3,1)=Z1(1
):F(2,1)=Z1(2)
1540 RETURN
1550 IF KE$<>"I"THEN 1690
1560 FOR Q=1 TO 3:Z(Q)=R(3,Q):Z1(Q)=B(1,Q):NEXT Q
1570 IF DIR<>1 THEN 1630
1580 FOR Q=1 TO 3
1590 R(3,Q)=T(4-Q,3):T(4-Q,3)=L(1,4-Q):L(1,4-Q)=BA
(Q,1):BA(Q,1)=Z(Q)
1600 NEXT Q
1610 B(1,1)=B(1,3):B(1,2)=B(2,3):B(1,3)=B(3,3):B(2
,3)=B(3,2):B(3,3)=B(3,1):B(3,2)=B(2,1):B(3,1)=Z1(1
):B(2,1)=Z1(2)
1620 RETURN
1630 FOR Q=1 TO 3
1640 Z1(Q)=B(Q,1)
1650 R(3,4-Q)=BA(4-Q,1):BA(4-Q,1)=L(1,Q):L(1,Q)=T(
Q,3):T(Q,3)=Z(4-Q)
1660 NEXT Q
1670 B(1,1)=B(3,1):B(2,1)=B(3,2):B(3,1)=B(3,3):B(3
,2)=B(2,3):B(3,3)=B(1,3):B(2,3)=B(1,2)
1680 B(1,3)=Z1(1):B(1,2)=Z1(2):RETURN
1690 FOR Q=1 TO 3:Z(Q)=R(2,Q):NEXT Q
1700 IF DIR<>1 THEN 1750
1710 FOR Q=1 TO 3
1720 R(2,Q)=T(4-Q,2):T(4-Q,2)=L(2,4-Q):L(2,4-Q)=BA
(Q,2):BA(Q,2)=Z(Q)
1730 NEXT Q
1740 RETURN
1750 FOR Q=1 TO 3
1760 R(2,4-Q)=BA(4-Q,2):BA(4-Q,2)=L(2,Q):L(2,Q)=T(
```

```
Q,2):T(Q,2)=Z(4-Q)
1770 NEXT Q
1780 RETURN
1790 REM **DRAW SCREEN**
1800 PEN 1:LOCATE 2,3:PRINT"THE CUBE"
1810 FOR I=1 TO 3
1820 FOR J=1 TO 3
1830 Q=F(I,J)
1840 X=(60+(I*44))
1850 FOR W=2 TO 39
1860 Y=W+30+(J*40)
1870 PLOT X,Y,Q:DRAWR 39,0,Q
1880 NEXT W
1890 NEXT J,I
1900 FOR I=1 TO 3
1910 FOR J=1 TO 3
1920 Q=T(I,J)
1930 Y=164+(J*28)
1940 FOR W=2 TO 36
1950 X=23+W+(I*46)+(J*37)
1960 IF J=3 THEN X=X-1
1970 PLOT X,Y,Q:DRAWR 28,24
1980 NEXT W
1990 NEXT J,I
2000 FOR I=1 TO 3
2010 FOR J=1 TO 3
2020 Q=R(I,J)
2030 FOR W=1 TO 32
2040 Y=(J*39)+W:Y=Y+3:IF I<>3 THEN Y=Y+(I*31) ELSE
 Y=Y+90
2050 IF I=1 THEN Y=Y+1
2060 X=202+(I*36)
2070 PLOT X,Y,Q:DRAWR 28,24
2080 NEXT W
2090 NEXT J,I
2100 FOR I=1 TO 3
2110 FOR J=1 TO 3
2120 Q=BA(I,J)
2130 X=(332+(I*44))
2140 FOR W=2 TO 39
2150 Y=W-40+(J*40)
2160 PLOT X,Y,Q:DRAWR 39,0,Q
2170 NEXT W
2180 NEXT J,I
2190 LOCATE 17,22:PRINT"BASE"
2200 FOR I=1 TO 3
2210 FOR J=1 TO 3
2220 Q=B(I,J)
2230 X=(332+(I*44))
2240 FOR W=2 TO 39
2250 Y=W+90+(J*40)
```

```
2260 PLOT X,Y,Q:DRAWR 39,0,Q
2270 NEXT W
2280 NEXT J,I
2290 LOCATE 17,14:PRINT"BACK"
2300 FOR I=1 TO 3
2310 FOR J=1 TO 3
2320 Q=L(I,J)
2330 X=(332+(I*44))
2340 FOR W=2 TO 39
2350 Y=W+220+(J*40)
2360 PLOT X,Y,Q:DRAWR 39,0,Q
2370 NEXT W
2380 NEXT J,I
2390 LOCATE 17,6:PRINT"LEFT"
2400 PLOT 70,95,1:TAG:PRINT"C";
2410 MOVE 70,135:TAG:PRINT"B";
2420 MOVE 70,175:TAG:PRINT"A";
2430 MOVE 115,60:TAG:PRINT"D";
2440 MOVE 157,60:TAG:PRINT"E";
2450 MOVE 200,60:TAG:PRINT"F";
2460 MOVE 244,78:TAG:PRINT"G";
2470 MOVE 278,105:TAG:PRINT"H";
2480 MOVE 310,128:TAG:PRINT"I";
2490 TAGOFF:RETURN
```

5. Crossword Puzzler

If you are one of those people who eagerly await the paper every morning just so that you can do the crossword, and, like most of us, get stuck and have to wait another twenty-four hours to get the result, then this program is for you.

The program will provide you with an endless supply of puzzles. If you happen to get stuck, then pressing the QUIT key will give you the solution immediately.

Playing instructions

When the game is loaded from tape and executed, a brief résumé of the playing instructions will appear on the screen and the database WORDS must be transferred into the computer. If you have followed the instructions for saving the programs correctly, then this should be situated on the tape after the main program.

When the loading process is completed, there will be a substantial delay during which the Amstrad organises its memory allocation. When this is completed you will be asked whether or not you require a time limit game (not for those with weak hearts!). When this has been decided, the crossword itself will be designed and the message MOVE CURSOR will appear in the appropriate window.

The game is played by using the cursor keys ↑ and ↓ to move a pointer up and down the list of clues on the left-hand side of the screen. When the required position is obtained, depressing the E key will cause the appropriate clue to appear and you can then input your solution.

During this input stage there are three options available:

1. Enter the answer by typing the appropriate letters followed by ENTER.
2. Delete the existing word by typing '–'.
3. End the game and see the solution: achieved by typing END.

Notes

1. If you are playing a 'time-limit' game the END or QUIT facility will not function and you must wait for your time to elapse before seeing the solution.
2. During the crossword design stage the message 'designing crossword' may flash, indicating that the program was unable to complete the puzzle due to the limitations of the database and a re-start is taking place.

It is important to remember that any crossword puzzle has one and only one solution. Even though it may appear that you have completed the puzzle, this is only the case if all twenty-seven of the individual words are correct.

The program

The sophistication of this program and the quality of the crosswords it produces are totally dependent on the size of the associated database which contains all the words and their respective clues. Accordingly, this chapter contains three separate programs:

1. PUZZLER
2. DATABASER1
3. DATABASER2

which should be entered and used in the correct sequence so that the program can operate properly.

To achieve this the programs should be entered and used as follows, for which you will require a blank cassette that can be used on both sides:

Step 1: Type in the main program PUZZLER and save this at the beginning of side 1

Step 2: Enter the second program DATABASER1 and type RUN. You will then be required to press 'play and record' on the Datacorder so that the database (file) called WORDS can be saved on side 1 of the tape after the PUZZLER program. It would be a good idea to note the reading on the tape counter at this point.

Step 3: Enter the third and final program DATABASER2 and save it on side 2 of the tape. This program can then be used

over and over again to increase the WORDS database, and instructions for this are included later in this chapter.

When you have performed these three steps and the WORDS database has been substantially extended the game is ready to be played.

PUZZLER

This is the main and longest program of the three and is used to design and manage the crossword as well as providing a user-machine interface.

Flowchart

The flowchart (opposite) represents the general operation of the program.

Variables

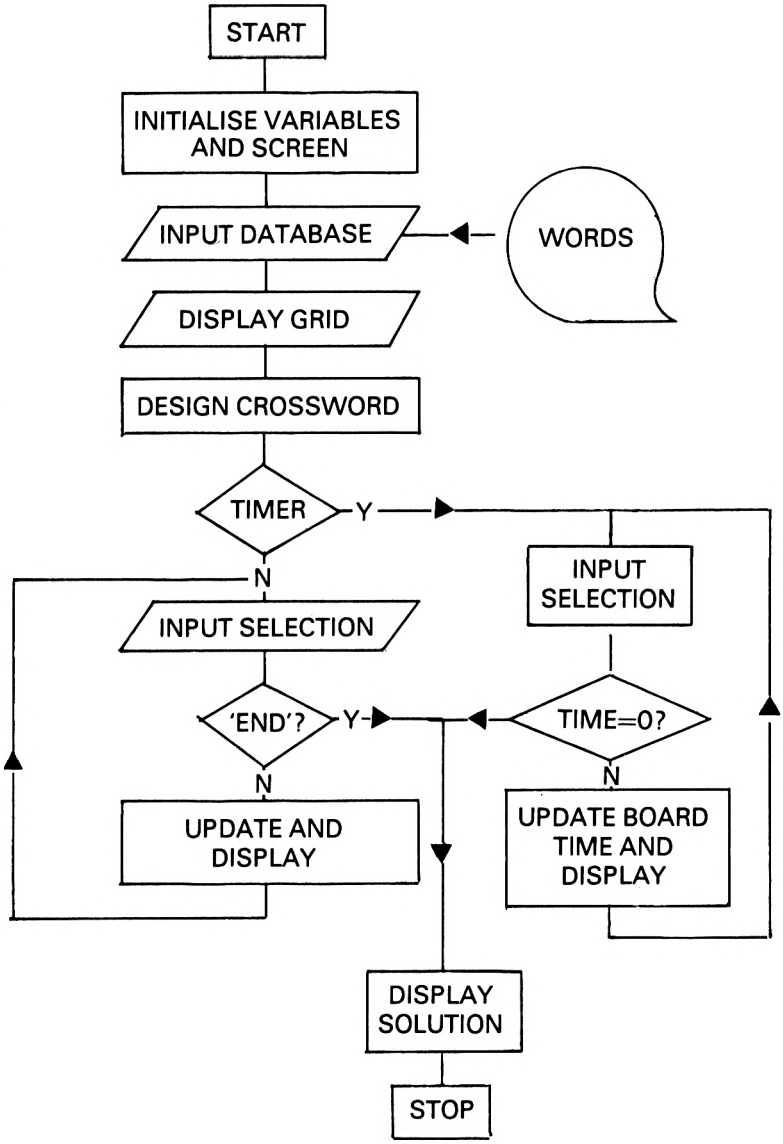
The following table represents the major variables used in the program and should help you to understand how the program operates.

A$()	Clue positions
C$(x,y)	Array containing information on grid
W$()	Words from database
CL$()	Clues from database
CN$()	Crossword clues
AN$()	Crossword answers
GUESS$	Your answer
CX	Horizontal position of pointer
CY	Vertical position of pointer

Comments on the program

Lines 10-20
Dimension arrays and call subroutine to display instructions on the screen.

Lines 30-160
This section opens a cassette input file and transfers the



information from the WORDS database into the computer where it is stored in two arrays W$() and CL$().

Lines 170-180
Initialises variables which are required in starting a new game.

Line 190
Due to a fault in the Amstrad's ROM, the handling of strings often causes excessive use of memory. This line checks to see if the amount of free memory falls below 2K. If so, a 'garbage collect' is performed resulting in an extra 13 or more K being available to the user.

Lines 200-300
This section of code initialises the array C$(14,14) used to hold the information for the puzzle. The elements are originally set to either 'O' indicating a white square into which a letter can be placed or a '*' indicating a black square.

Lines 310-870
This is the main section used to create the graphical representation on the screen.

Lines 880-1020
This is the control section for the design of the puzzle and it makes use of the routines at 1790, 2270 and 2530 to produce a completed crossword from the information in the database.

Lines 1030-1070
This is used to initialise the timer if it is required, with full use of the interrupt facilities as described in Chapter 2.

Lines 1080-1210
The screen display is completed with the clues and pointer being shown on the screen.

Lines 1220-1600
The main program, which controls the solving of the puzzle and legality checks, etc.

Lines 1610-1780
This is the routine used to set A$(), which contains the positional information from the crossword clues. The information is held in a four-character alphanumeric string. For example,

```
A$(1)=DB,A3
```

where D = down
B,A = starting position on grid
3 = size of word

Lines 1790-2040
This is one of the three artificial intelligence routines in the program. It is used to locate a word and clue to fit in the horizontal positions of the grid.

Lines 2050-2260
This routine is used to print the current situation on the grid. It is used throughout the program to display the intermediate positions and at the end to display the final solution.

Lines 2270-2520
Similar to the routine at 1790, this is used for the vertical positions on the grid.

Lines 2530-2620
This is the third and final routine used in the formation of the crossword puzzle, where the computer checks that the combinations used do not include any repetition.

Lines 2630-3030
This routine controls the movement of the pointer and allows the user to input his answer. The answer is then checked for type ('END', '-' or a word), a legality test is performed and the appropriate action is taken.

Lines 3040-3060
A short routine used to print error messages when an entry fails to pass a legality check.

Lines 3070-3240
This checks for a correct solution to the crossword by comparing the contents of the array C$() with COPY$().

Lines 3250-3610
A very complex routine which is called when the computer is required to remove an already positioned word. This is very complicated as any letters which are part of another word must not be erased. For example, on deleting HOUSE in the following the letter S must not be removed:

```
    A
HOUSE
    H
```

This is an excellent example of pattern recognition.

Lines 3620-3800
The interrupt routine, used to control the on-screen timer if it is required.

Lines 3810-3920
A routine called very early in the program, which prints a brief résumé of the instructions on the screen.

The listing

The game should be entered and played in the caps lock mode. All spaces within the text should be entered as they appear.

```
10 DIM A$(27),CS$(40,2),C$(14,14),COPY$(14,14),W$(
1000),CL$(1000),CN$(27),AN$(27)
20 GOSUB 3810
30 OPENIN "WORDS":LET I=1
40 INPUT#9,W$(I)
50 IF W$(I)="XXX" THEN LET I=I-1:GOTO 80
60 INPUT#9,CL$(I)
70 LET I=I+1:GOTO 40
80 LET WORDS=I-1
90 LET I=1
100 IF EOF THEN GOTO 150
110 INPUT #9,CS$(I,1)
120 INPUT #9,CS$(I,2)
130 LET I=I+1
140 GOTO 100
150 CLOSEIN
160 LET CLUES=I-1
170  RESTORE:LET GOVER=0
180 LET NW=1
190 IF FRE(0)<2048 THEN LET D=FRE("")
200 LET CY=3:LET CX=1
210 FOR I=1 TO 13
220 FOR J=1 TO 13
230 READ Z$
240 LET C$(J,I)=Z$
250 NEXT J,I
260 DATA 0,0,0,0,0,0,0,0,*,0,0,0,0,*,0,*,*,*,*,*,0
,*,0,*,*,0,0,0,0,0,0,*,0,0,0,0,0,*,0,0,*,*,*,0,*,*
,0,*,0,*,*,0
270 DATA 0,0,0,0,0,0,0,*,*,*,0,*,*,0,*,*,*,0,*,0,0
,0,0,0,*,0,0,*,*,*,*,*,0,*,0,*,0,*,0,*,0,0,0,0,0,0
,*,0,*,0,*,0
280 DATA 0,*,*,0,*,*,0,*,0,*,0,*,0,0,0,0,0,0,0,0,*,*
,*,0,0,0,0,0,*,0,*,0,*,0,*,0,*,*,*,0,0,*,0,*,0,*,0
,*,0,0,0,*,0
```

```
290 DATA 0,*,0,*,0,0,0,0,0,*,*,*,0
300 FOR I=0 TO 14:LET C$(I,0)="*":LET C$(I,14)="*"
:LET C$(0,I)="*":LET C$(14,I)="*":NEXT I
310 GOSUB 1610
320 INK 0,26
330 INK 1,0
340 INK 2,21
350 INK 3,15
360 PEN#2,3
370 PAPER 2
380 IF FAIL=0 THEN CLS ELSE LOCATE 1,1
390 WINDOW#3,3,13,20,24
400 PAPER#3,0:CLS#3:PEN#3,3:LOCATE#3,1,2:PRINT#3,"
 TIME LEFT":PEN#3,1:LOCATE#3,4,3:PRINT#3,"10:00"
410 WINDOW#2,15,38,20,24
420 PAPER#2,0:CLS#2
430 LET GAME=1:GOSUB 1080
440 PAPER 0
450 PEN 1
460 LET J=0
470 FOR Y=370 TO 150 STEP -18
480 LET I=0
490 LET J=J+1
500 FOR X=350 TO 570 STEP 18
510 LET I=I+1
520 PLOT 0,0,1
530 MOVE X,Y
540 TAG
550 IF C$(I,J)="*" THEN PRINT CHR$(143);:GOTO 590
560 PLOT 0,0,2
570 MOVE X,Y
580 PRINT CHR$(32);
590 TAGOFF
600 NEXT X
610 NEXT Y
620 FOR I=349 TO 589 STEP 18
630 MOVE I,140
640 DRAWR 0,230,1
650 NEXT I
660 FOR I=372 TO 130 STEP -18
670 MOVE 349,I
680 DRAWR 234,0,1
690 NEXT I
700 PLOT 0,0,1
710 LET X=0
720 FOR I=350 TO 570 STEP 18
730 LET X=X+1
740 MOVE I,390
750 TAG
760 PRINT CHR$(64+X);
770 TAGOFF
```

```
780 NEXT I
790 LET Y=0
800 FOR J=370 TO 138 STEP -18
810 LET Y=Y+1
820 MOVE 328,J
830 TAG
840 PRINT CHR$(64+Y);
850 TAGOFF
860 NEXT J
870 LOCATE#2,3,2
880 PRINT#2, "DESIGNING CROSSWORD"
890 FOR P=1 TO 27
900 RANDOMIZE (TIME)
910 LET ST=INT(RND*500)+1
920 LET DIR$=LEFT$(A$(P),1)
930 IF DIR$="A" THEN LET S=ST:LET F=WORDS:GOSUB 17
90
940 IF DIR$="A" AND TES=0 THEN LET S=1:LET F=ST:GO
SUB 1790
950 IF FAIL=1 AND S=1 THEN GOTO 170
960 LET TES=0:LET FAIL=0
970 IF DIR$="D" THEN LET S=ST:LET F=WORDS:GOSUB 22
70
980 IF DIR$="D" AND TES=0 THEN LET S=1:LET F=ST:GO
SUB 2270
990 IF FAIL=1 AND S=1 THEN GOTO 170
1000 LET TES=0:LET FAIL=0
1010 NEXT P
1020 RESTORE:FOR J=1 TO 13:FOR I=1 TO 13:LET COPY$
(I,J)=C$(I,J):READ C$(I,J):NEXT I,J
1030 LET MI=10:LET SEC=0
1040 CLS#2:LOCATE#2,6,2:PRINT#2,"DO YOU WANT":LOCA
TE#2,5,3:PRINT#2,"A TIME LIMIT ?"
1050 LET Z$=INKEY$:IF Z$="" THEN GOTO 1050
1060 IF Z$="Y" THEN EVERY 50,1 GOSUB 3620:LET LIMI
T=1:GOTO 1080
1070 IF Z$="N" THEN LET LIMIT=0:LET D=REMAIN(1):CL
S#3:PEN#3,3:LOCATE#3,4,2:PRINT#3,"TIMER":LOCATE#3,
1,3:PRINT#3,"INOPERATIVE":LOCATE#3,1,5:PEN#3,1:PRI
NT#3,"END";:PEN#3,3:PRINT#3," TO QUIT" ELSE GOTO 1
050
1080 IF GAME=0 THEN GOTO 1230
1090 PAPER 2
1100 ZONE 8
1110 PRINT "  DOWN"," ACROSS"
1120 PRINT "  ----"," ------"
1130 FOR I=1 TO 27
1140 IF LEFT$(A$(I),1)="D" THEN PRINT " ";MID$(A$(
I),2,2);" (";RIGHT$(A$(I),1);")"
1150 NEXT I
1160 LET DO=3
```

```
1170 LOCATE CX,CY:PRINT CHR$(243)
1180 FOR I=1 TO 27
1190 LOCATE 10,DO
1200 IF LEFT$(A$(I),1)="A" THEN PRINT MID$(A$(I),2
,2);" (";RIGHT$(A$(I),1);")":LET DO=DO+1
1210 NEXT I
1220 IF GAME=1 THEN LET GAME=0:RETURN
1230 GOSUB 2630
1240 IF GOVER=1 THEN GOTO 3680
1250 IF GUESS$="END" AND LIMIT=0 THEN GOTO 3690
1260 IF GUESS$="-" THEN GOTO 3250
1270 IF LEN(GUESS$)<>LEN(AN$(PE))THEN GOTO 3040
1280 LET X$=MID$(A$(PE),2,1)
1290 LET Y$=MID$(A$(PE),3,1)
1300 LET X=ASC(X$)-64
1310 LET Y=ASC(Y$)-64
1320 LET L=LEN(GUESS$)
1330 LET K=1
1340 IF LEFT$(A$(PE),1)="D" THEN GOTO 1470
1350 FOR N=X TO X+L-1
1360 IF C$(N,Y)="0" THEN GOTO 1390
1370 IF C$(N,Y)=MID$(GUESS$,K,1) THEN GOTO 1390
1380 GOTO 3040
1390 LET K=K+1:NEXT N
1400 LET K=1
1410 FOR J=X TO X+L-1
1420 LET C$(J,Y)=MID$(GUESS$,K,1)
1430 LET K=K+1
1440 NEXT J
1450 GOSUB 2050
1460 GOTO 3070
1470 FOR N=Y TO Y+L-1
1480 IF C$(X,N)="0" THEN GOTO 1510
1490 IF C$(X,N)=MID$(GUESS$,K,1) THEN GOTO 1510
1500 GOTO 3040
1510 LET K=K+1:NEXT N
1520 LET K=1
1530 FOR J=Y TO Y+L-1
1540 LET C$(X,J)=MID$(GUESS$,K,1)
1550 LET K=K+1
1560 NEXT J
1570 GOSUB 2050
1580 GOTO 3070
1590 GOTO 1590
1600 END
1610 RESTORE 1650
1620 FOR I=1 TO 27
1630 READ A$(I)
1640 NEXT I
1650 DATA AAA8,DBA3
1660 DATA AAC5,DEC4
```

```
1670 DATA AAE7,DAC5
1680 DATA DHA4,AGC5
1690 DATA DJA4,AJA4
1700 DATA DMA4,DGE5
1710 DATA ABH6,DDH3
1720 DATA AAJ6,DAI5
1730 DATA DCJ4,DEJ4
1740 DATA AEM5,DGK3
1750 DATA DIK3,AIL3,AGF5
1760 DATA DIF4,DKE6
1770 DATA AJJ4,DMF8
1780 RETURN
1790 REM DESIGN HORIZONTALS
1800 LET X$=MID$(A$(P),2,1)
1810 LET Y$=MID$(A$(P),3,1)
1820 LET X=ASC(X$)-64
1830 LET Y=ASC(Y$)-64
1840 LET L=VAL(RIGHT$(A$(P),1))
1850 FOR M=S TO F
1860 IF LEN(W$(M))<>L THEN GOTO 1960
1870 LET K=1
1880 FOR N=X TO X+L-1
1890 IF C$(N,Y)="0" THEN GOTO 1920
1900 IF C$(N,Y)=MID$(W$(M),K,1) THEN GOTO 1920
1910 GOTO 1960
1920 LET K=K+1:NEXT N
1930 GOSUB 2530
1940 IF OK=0 THEN GOTO 1960
1950 GOTO 1980
1960 NEXT M
1970 IF S<>1 THEN GOTO 2040 ELSE LET FAIL=1:GOTO 2
040
1980 LET K=1
1990 FOR J=X TO X+(L-1)
2000 LET C$(J,Y)=MID$(W$(M),K,1)
2010 LET K=K+1
2020 NEXT J
2030 LET TES=1
2040 RETURN
2050 REM ***TO BE REMOVED***
2060 PRINT CHR$(22)+CHR$(1);
2070 PAPER 0
2080 LET J=0
2090 FOR Y=370 TO 150 STEP -18
2100 LET I=0
2110 LET J=J+1
2120 FOR X=350 TO 570 STEP 18
2130 LET I=I+1
2140 PLOT 0,0,1
2150 MOVE X,Y
2160 TAG
```

```
2170 IF C$(I,J)="*" THEN PRINT CHR$(143);:GOTO 221
0
2180 IF C$(I,J)="0" THEN PRINT CHR$(32);: GOTO 221
0
2190 MOVE X,Y
2200 PRINT C$(I,J);
2210 TAGOFF
2220 NEXT X
2230 NEXT Y
2240 PRINT CHR$(22)+CHR$(0);
2250 CLS#2
2260 RETURN
2270 REM DESIGN VERTICALS
2280 LET X$=MID$(A$(P),2,1)
2290 LET Y$=MID$(A$(P),3,1)
2300 LET X=ASC(X$)-64
2310 LET Y=ASC(Y$)-64
2320 LET L=VAL(RIGHT$(A$(P),1))
2330 FOR M=S TO F
2340 IF LEN(W$(M))<>L THEN GOTO 2440
2350 LET K=1
2360 FOR N=Y TO Y+L-1
2370 IF C$(X,N)="0" THEN GOTO 2400
2380 IF C$(X,N)=MID$(W$(M),K,1) THEN GOTO 2400
2390 GOTO 2440
2400 LET K=K+1:NEXT N
2410 GOSUB 2530
2420 IF OK=0 THEN GOTO 2440
2430 GOTO 2460
2440 NEXT M
2450 IF S<>1 THEN GOTO 2520 ELSE LET FAIL=1:GOTO 2
520
2460 LET K=1
2470 FOR J=Y TO Y+(L-1)
2480 LET C$(X,J)=MID$(W$(M),K,1)
2490 LET K=K+1
2500 NEXT J
2510 LET TES=1
2520 RETURN
2530 REM **CHECK FOR REPETITION**
2540 LET OK=1
2550 FOR JK=1 TO NW-1
2560 IF AN$(JK)=W$(M) THEN LET OK=0:RETURN
2570 NEXT JK
2580 LET OK=1
2590 LET AN$(NW)=W$(M)
2600 LET CN$(NW)=CL$(M)
2610 LET NW=NW+1
2620 RETURN
2630 REM **CURSOR**
2640 CLS#2
```

```
2650 LOCATE#2,6,2
2660 PRINT#2,"MOVE CURSOR"
2670 PAPER 2
2680 LOCATE CX,CY
2690 FOR DEL=1 TO 100:NEXT
2700 IF INKEY(2)=0 THEN CY=CY+1:PRINT" "
2710 IF CY=19 AND CX=1 THEN CY=3:CX=9
2720 IF CY=14 AND CX=9 THEN CY=3:CX=1
2730 IF INKEY(0)=0 THEN CY=CY-1:PRINT" "
2740 IF CY=2 AND CX=1 THEN CY=13:CX=9
2750 IF CY=2 AND CX=9 THEN CY=18:CX=1
2760 IF INKEY(58)=0 THEN GOTO 2810
2770 LOCATE CX,CY
2780 PRINT CHR$(243)
2790 IF GOVER=1 THEN GOTO 3680
2800 GOTO 2680
2810 CLS#2:FOR LM=1 TO 20
2820 LET Z$=INKEY$
2830 NEXT LM
2840 LET PE=CY-2
2850 IF CX=1 THEN GOTO 2890
2860 FOR DE=1 TO 27
2870 IF LEFT$(A$(DE),1)="A" THEN LET PE=PE-1:IF PE
=0 THEN LET PE=DE:GOTO 2920
2880 NEXT DE
2890 FOR DE=1 TO 27
2900 IF LEFT$(A$(DE),1)="D" THEN LET PE=PE-1:IF PE
=0 THEN LET PE=DE:GOTO 2920
2910 NEXT DE
2920 PEN#2,1:LOCATE#2,1,1:PRINT#2,"THE CLUE IS:-"
2930 PEN#2,3
2940 IF LEN(CN$(PE))>2 THEN LOCATE#2,INT((23-LEN(C
N$(PE)))/2),2:PRINT#2,CN$(PE):GOTO 2990
2950 FOR JH=1 TO CLUES
2960 IF CS$(JH,1)=CN$(PE) THEN GOTO 2980
2970 NEXT JH
2980 LOCATE#2,INT((23-LEN(CS$(JH,2)))/2),2:PRINT#2
,CS$(JH,2)
2990 PEN#2,1
3000 LOCATE#2,1,3:PRINT#2,"ENTER WORD (- TO DELETE
)"
3010 PEN#2,3
3020 LOCATE#2,1,4:INPUT#2,"",GUESS$
3030 RETURN
3040 REM **PRINT ERROR MESSAGE**
3050 CLS#2
3060 GOTO 1080
3070 REM **CHECK FOR SOLUTION **
3080 FOR I=1 TO 13
3090 FOR J=1 TO 13
3100 IF C$(I,J)<>COPY$(I,J) THEN GOTO 1080
```

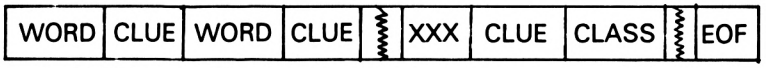
```
3110 NEXT J
3120 NEXT I
3130 LET D=REMAIN(1)
3140 FOR J=1 TO 10
3150 FOR I=1 TO 100
3160 INK 1,INT(RND*25)+1
3170 SOUND 1,I*10,1:SOUND 2,I/5,1
3180 IF SQ(1)<>4 THEN GOTO 3180
3190 NEXT I
3200 NEXT J
3210 INK 1,0
3220 CLS#2
3230 PEN#2,1
3240 LOCATE#2,4,3:PRINT#2,"PUZZLE COMPLETE":GOTO 3
760
3250 REM **REMOVE WORD**
3260 X$=MID$(A$(PE),2,1)
3270 Y$=MID$(A$(PE),3,1)
3280 LET X=ASC(X$)-64
3290 LET Y=ASC(Y$)-64
3300 CLS#2
3310 IF LEFT$(A$(PE),1)="D" THEN GOTO 3470
3320 FOR L=X TO X+LEN(AN$(PE))-1
3330 FOR H=1 TO 8
3340 IF C$(L,Y-H)="*" THEN GOTO 3370
3350 IF C$(L,Y-H)="0" THEN GOTO 3420
3360 NEXT H
3370 FOR HH=1 TO 8
3380 IF C$(L,Y+HH)="*" THEN GOTO 3430
3390 IF C$(L,Y+HH)="0" THEN GOTO 3420
3400 NEXT HH
3410 GOTO 3430
3420 LET C$(L,Y)="0"
3430 IF H=1 AND HH=1 THEN LET C$(L,Y)="0"
3440 NEXT L
3450 GOSUB 2050
3460 GOTO 1080
3470 FOR L=Y TO Y+LEN(AN$(PE))-1
3480 FOR H=1 TO 8
3490 IF C$(X-H,L)="*" THEN GOTO 3520
3500 IF C$(X-H,L)="0" THEN GOTO 3570
3510 NEXT H
3520 FOR HH=1 TO 8
3530 IF C$(X+HH,L)="*" THEN GOTO 3580
3540 IF C$(X+HH,L)="0" THEN GOTO 3570
3550 NEXT HH
3560 GOTO 3580
3570 LET C$(X,L)="0"
3580 IF H=1 AND HH=1 THEN LET C$(X,L)="0"
3590 NEXT L
3600 GOSUB 2050
```

```
3610 GOTO 1080
3620 REM**TIMER**
3630 LET SEC=SEC-1:IF SEC=-1 THEN LET SEC=59:LET M
I=MI-1
3640 PEN#3,1:LOCATE#3,4,3:PRINT#3," ";USING "#";MI
;:PRINT#3,":";:PRINT#3,USING "##";SEC
3650 IF SEC<10 THEN LOCATE#3,7,3:PRINT#3,"0"
3660 IF MI=0 AND SEC=0 THEN LET D=REMAIN(1):LET GO
VER=1
3670 RETURN
3680 CLS#2:LOCATE#2,7,2:PRINT#2,"OUT OF TIME"
3690 FOR I=1 TO 13
3700 FOR J=1 TO 13
3710 LET C$(I,J)=COPY$(I,J)
3720 NEXT J
3730 NEXT I
3740 GOSUB 2050
3750 CLS#2
3760 PEN#2,3:LOCATE#2,6,2:PRINT#2,"PLAY AGAIN ?"
3770 LET Z$=INKEY$:IF Z$="" THEN GOTO 3770
3780 IF Z$="Y" THEN GOTO 170
3790 IF Z$="N" THEN PAPER 2:PEN 1:LOCATE 1,24:END
3800 GOTO 3770
3810 REM **START**
3820 BORDER 1:INK 0,21:INK 1,0:INK 2,21
3830 PAPER 0
3840 CLS
3850 LOCATE 15,2:PRINT"WELCOME  TO"
3860 LOCATE 12,4:INK 3,6:PEN 3:PRINT"CROSSWORD PUZ
ZLER"
3870 PEN 1
3880 LOCATE 1,8
3890 PRINT" THE OBJECT OF THIS GAME IS TO COMPLETE
 A CROSSWORD THAT WILL BE DISPLAYED ON   THE SCREE
N."
3900 PRINT" IN ORDER TO SEE A CLUE, MOVE THE CURSO
RUSING THE ARROW KEYS UNTIL  IT IS BESIDETHE COORD
INATES OF THE CLUE NEEDED, THENDEPRESS THE 'E' KEY
. A CLUE WILL THEN BEDISPLAYED AND YOU WILL BE ASK
ED TO PLACEA WORD."
3910 PRINT" IF YOU DO NOT WISH TO PLACE A WORD,
 DEPRESS THE ENTER KEY. IF YOU WISH TO   CLEAR A W
ORD PLACED IN THE CROSSWORD BY MISTAKE, ENTER '-'.
3920 LOCATE 8,22:PRINT"ENTER DATABASE CASSETTE ":R
ETURN
```

DATABASER1

As we have indicated, this program requires a substantial database to operate well and we therefore require two programs to create and then manage the file.

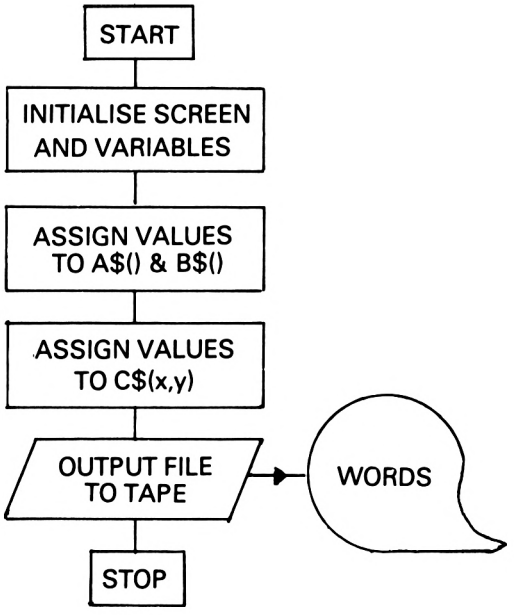
The first of these is a short write only program (see Chapter 2), which produces a very short file with a structure as indicated below:



The delimiter XXX is used to separate the two sections of the file, the first part of which contains the words and clues, and the second part the classifications for the two-letter clues. These are used to save on the amount of memory required.

Flowchart

This flowchart represents the general operation of the program:



Variables

A$()	Array of words
B$()	Array of clues
C$(x,y)	Array containing classifications

Comments on the program

The program is short and easy to follow. The principles of its operation have been outlined in Chapter 2.

The listing

```
10 DIM A$(20),B$(20),CS$(5,2)
20 CLS
30 LOCATE 12,1:PRINT"DATABASER.1"
40 LOCATE 12,2:PRINT"----------"
50 FOR I=1 TO 15
60 READ A$(I),B$(I)
70 NEXT I
80 FOR I=1 TO 2
90 READ CS$(I,1),CS$(I,2)
100 NEXT I
110 OPENOUT "WORDS"
120 FOR J=1 TO 15
130 PRINT#9,A$(J)
140 PRINT#9,B$(J)
150 NEXT J
160 PRINT#9,"XXX"
170 FOR J=1 TO 2
180 PRINT#9, CS$(J,1)
190 PRINT#9, CS$(J,2)
200 NEXT J
210 CLOSEOUT
220 END
230 DATA LION,AN,TIGER,AN,ELEPHANT,AN,ZEBRA,AN,HOR
SE,AN,DONKEY,AN,GIRAFFE,AN,COW,AN,PIG,AN
240 DATA MONKEY,AN,GORILLA,AN,LONDON,ET,YORK,ET,CA
RDIFF,ET,DOVER,ET
250 DATA AN,ANIMAL,ET,ENGLISH TOWN
```

DATABASER2

The WORDS file created so far is of no practical use as it only contains fifteen words from which it would be impossible to design a complete crossword.

This second program is therefore of the utmost importance as it allows the database to be extended, and it can be used as often as required.

When the program is executed the WORDS file will be loaded from tape, and after the usual substantial delay the first word and clue:

LION AN

will be displayed on the screen. The program is then in 'edit mode' during which time the whole file will be displayed one record at a time and you can either pass on to the next record by pressing the space bar, or edit the record by pressing the E key. When the alteration is complete, the normal sequence will continue until all the existing records have been displayed.

On completion of this routine the program will enter 'input mode' so that more words and clues can be declared. In each case the clue can be a two-letter classification or a word or short sentence, and a good selection of possibilities is listed at the end of this chapter.

To exit from the 'input mode' we simply type END for the word. The computer than asks for any new classifications to be entered and the new file is saved on to tape in place of the old version, with the tape positioned directly after the main program.

Flowchart

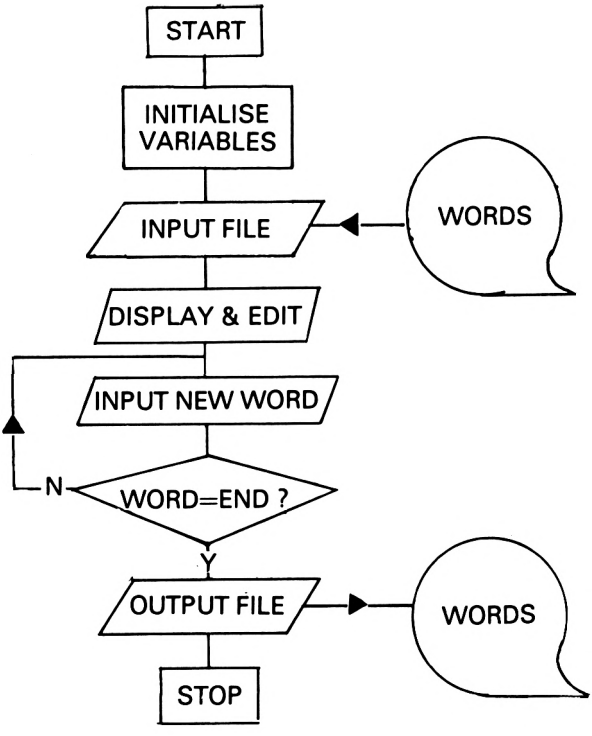
The flowchart on p.90 represents the general operation of the program.

Variables

A$()	Array of words
B$()	Array of clues
C$(x,y)	Array containing classifications
I	Word counter
CLUES	Classification counter

Comments on the program

Lines 10-20
Initialise variables and dimension arrays.



Lines 30-150
Input information currently stored on tape into the computer.

Lines 160-250
Display and edit mode where corrections can be made to the current database.

Lines 260-350
Input section where new words and clues can be entered into the database.

Lines 360-480
The two-letter classifications are checked and if any are missing then the computer requests information regarding these clues.

Lines 490-590
The new updated database is saved on to tape.

The listing

```
10 DIM A$(1000),B$(1000),CS$(40,2)
20 LET I=1
30 OPENIN "WORDS"
40 INPUT #9,A$(I)
50 IF A$(I)="XXX" THEN GOTO 90
60 INPUT #9,B$(I)
70 LET I=I+1
80 GOTO 40
90 LET CLUES=1
100 IF EOF THEN GOTO 150
110 INPUT #9,CS$(CLUES,1)
120 INPUT #9,CS$(CLUES,2)
130 LET CLUES=CLUES+1
140 GOTO 100
150 CLOSEIN
160 FOR K=1 TO I-1
170 PRINT A$(K),B$(K),LEN(A$(K))
180 LET Q$=INKEY$:IF Q$="" THEN GOTO 180
190 IF Q$=" " THEN GOTO 250
200 IF K=I THEN GOTO 260
210 CLS
220 INPUT "WORD=";A$(K)
230 INPUT "CATEGORY=";B$(K)
240 K=K-1
250 NEXT K
260 CLS
270 PRINT "COUNT=";I
280 INPUT "WORD=";A$(I)
290 IF A$(I)="END" THEN GOTO 360
300 FOR J=1 TO I-1
310 IF A$(J)=A$(I) THEN GOTO 260
320 NEXT J
330 INPUT "CATEGORY=";B$(I)
340 LET I=I+1
350 GOTO 260
360 CLS
370 PRINT "CATEGORIES"
380 PRINT "----------"
390 FOR J=1 TO I-1
400 IF LEN(B$(J))>2 THEN GOTO 480
410 FOR K=1 TO CLUES
420 IF CS$(K,1)=B$(J) THEN GOTO 480
430 NEXT K
440 LET CS$(CLUES,1)=B$(J)
450 PRINT "GROUP=";B$(J)
460 INPUT "CLASS=";CS$(CLUES,2)
470 LET CLUES=CLUES+1
480 NEXT J
490 OPENOUT "WORDS"
```

```
500 FOR J=1 TO I-1
510 PRINT#9,A$(J)
520 PRINT#9,B$(J)
530 NEXT J
540 PRINT#9,"XXX"
550 FOR J=1 TO CLUES-1
560 PRINT#9, CS$(J,1)
570 PRINT#9, CS$(J,2)
580 NEXT J
590 CLOSEOUT
```

Clues

The following few pages contain a selection of words and clues which can be used to increase the database.

Remember: the larger the database, the more variety in the puzzles.

Words with classifications

BRISTOL	ET	CAMEL	AN
PLAICE	FI	WOLF	AN
CARP	FI	ESSEX	EC
STINGRAY	FI	VAN	EC
DEER	AN	MOLE	TR
RABBIT	AN	KANGAROO	AN
KENT	EC	CORNWALL	EC
MOUSE	AN	SOMERSET	EC
MOOSE	AN	LLAMA	AN
PANDA	AN	NORFOLK	EC
SHARK	FI	GUITAR	MI
SALMON	FI	VIOLIN	MI
ROACH	FI	BASSOON	MI
TENCH	FI	PANSY	FL
FOX	AN	HARP	MI
HARE	AN	ASS	AN
SUFFOLK	EC	OXFORD	ET
BADGER	AN	CLARINET	MI
ANTEATER	AN	ORANGE	FR
DEVON	EC	GRAPE	FR
COD	FI	APRICOT	FR
PIKE	FI	BEAR	AN
HADDOCK	FI	OSTRICH	BI

GREEN	CO	SILVER	EL
MAGENTA	CO	ZINC	EL
BLACK	CO	OXYGEN	EL
LYNX	AN	EXETER	ET
SPIDER	IN	PUMA	AN
GIBBON	AN	TRUMPET	MI
HERRING	FI	ORGAN	MI
EARWIG	IN	CELLO	MI
CHINA	WC	ROSE	FL
TIBET	WC	DAISY	FL
POLAND	WC	HYENA	AN
FINLAND	WC	PENGUIN	BI
EGYPT	WC	BANJO	MI
BELGIUM	WC	OBOE	MI
ITALY	WC	BANANA	FR
YEMEN	WC	ABELE	TE
GOLF	SP	LEMON	FR
BOWLS	SP	NORWICH	ET
HOLLAND	WC	TROUT	FI
TENNIS	SP	YELLOW	CO
SNOOKER	SP	WHITE	CO
SLOTH	AN	SQUIRREL	AN
POPPY	FL	FLEA	IN
SKIING	SP	BAGPIPES	MI
PLUMB	FR	BUFFALO	AN
YEN	CU	FRANCE	WC
LIRA	CU	INDIA	WC
BACH	GC	PAKISTAN	WC
MOZART	GC	BULGARIA	WC
TOMATO	FR	SWEDEN	WC
LETTUCE	VE	CHILE	WC
SWEDE	VE	SPAIN	WC
BIZET	GC	GREECE	WC
LEEDS	ET	FOOTBALL	SP
TOAD	AN	BASEBALL	SP
COLUMBIA	WC	AMERICA	WC
JUDO	SP	HOCKEY	SP
FLOUNDER	FI	SQUASH	SP
MAHLER	GC	ASIA	CT
HAYDN	GC	EUROPE	CT
BERLIN	CC	RUGBY	SP
MADRID	CC	WORM	AN
BRUSSELS	CC	KOREA	WC
MOSCOW	CC	LIBYA	WC
CARBON	EL	DOLLAR	CU

BRAHMS	GC	JAPAN	WC
ROSSINI	GC	NORWAY	WC
POTATO	VE	ICELAND	WC
BEAN	VE	PERU	WC
ONION	VE	PORTUGAL	WC
PARSNIP	VE	ISRAEL	WC
LISZT	GC	CRICKET	SP
HAMSTER	AN	BOXING	SP
BLUE	CO	AUSTRIA	WC
SHOOTING	SP	LACROSSE	SP
SWIMMING	SP	DARTS	SP
STRAUSS	GC	TAPIR	AN
SCHUBERT	GC	AFRICA	CT
GRIEG	GC	SAILING	SP
BONN	CC	FENCING	SP
ATHENS	CC	LEMUR	AN
TRIPOLI	CC	ETHIOPIA	WC
PEKING	CC	FRANC	CU
IRON	EL	WALTON	GC
HYDROGEN	EL	BRITTEN	GC
TIN	EL	CARROT	VE
TITANIUM	EL	MARROW	VE
GOAT	AN	CABBAGE	VE
CHEETAH	AN	VIOLA	MI
DRUM	MI	CHERRY	FR
PIANO	MI	FROG	AN
FLUTE	MI	BRAZIL	WC
TULIP	FL	SKATING	SP
BISON	AN	PERCH	FI
SEALION	AN	WAGNER	GC
CYMBAL	MI	DVORAK	GC
APPLE	FR	PARIS	CC
PEACH	FR	LISBON	CC
MELON	FR	ROME	CC
LIME	FR	TOKYO	CC
REINDEER	AN	CANBERRA	CC
RED	CO	GOLD	EL
PINK	CO	HELIUM	EL
CYAN	CO	MERCURY	EL
ANT	IN	COPPER	EL
BEETLE	IN	MAGNESIUM	EL
BEE	IN	BARIUM	EL
MOTH	IN	URANIUM	EL
GERMANY	WC	NIGERIA	WC
RUSSIA	WC	MOON	AL

VENUS	AL	PLATYPUS	AN
JUPITER	AL	BRACES	CL
URANUS	AL	PILCHARD	FI
COMET	AL	ROOK	BI
UNIVERSE	AL	FALCON	BI
SHOE	CL	SPARROW	BI
JACKET	CL	KESTREL	BI
SCARF	CL	WREN	BI
SHIRT	CL	OFFICE	BU
SHORTS	CL	BEECH	TE
KNICKERS	CL	MOSQUITO	IN
GLOVE	CL	TIT	BI
CUP	KU	LYRE	MI
MUG	KU	LIZARD	AN
SPOON	KU	SILICON	EL
PAN	KU	RHODIUM	EL
DRAINER	KU	WARSAW	CC
LEG	HA	UGANDA	WC
HAND	HA	SUN	AL
TOE	HA	EARTH	AL
BACK	HA	SATURN	AL
THROAT	HA	PLUTO	AL
BREAST	HA	NEBULA	AL
QUASAR	AL	METEOR	AL
LACES	CL	SOCK	CL
NERVES	HA	COAT	CL
GUT	HA	TROUSERS	CL
ROE	AN	JUMPER	CL
SPIT	KU	JEANS	CL
GOOSE	BI	BRA	CL
PLANE	TR	MITTEN	CL
SHIP	TR	GLASS	KU
TAXI	TR	KNIFE	KU
FLAT	BU	DISH	KU
SHOP	BU	SAUCEPAN	KU
MUSEUM	BU	MOUTH	HA
SCHOOL	BU	FOOT	HA
HOTEL	BU	FINGER	HA
KEEP	BU	KNEE	HA
LADLE	KU	STOMACH	HA
TEA	BE	TONGS	KU
LAGER	BE	PULSAR	AL
PORT	BE	TONGUE	HA
WHISKY	BE	BRAIN	HA
ADVOCAAT	BE	MITE	IN

NEON	EL	HAT	CL
SWAN	BI	PANTS	CL
CAR	TR	VEST	CL
JET	TR	CARDIGAN	CL
BALLOON	TR	PLATE	KU
CHURCH	BU	FORK	KU
HUT	BU	POT	KU
LIBRARY	BU	AARDVARK	AN
BUNGALOW	BU	ARM	HA
COLLEGE	BU	NOSE	HA
MANSION	BU	ANKLE	HA
CABIN	BU	THUMB	HA
MOSQUE	BU	BUM	HA
COFFEE	BE	CHEST	HA
CIDER	BE	BIN	KU
WINE	BE	TIE	CL
SHERRY	BE	SPINE	HA
COLA	BE	TONSILS	HA
KILT	CL	HANDEL	GC
GARTER	CL	FEZ	CL
TUNA	FI	DUCK	BI
GULL	BI	BUS	TR
ROBIN	BI	TRAIN	TR
PANTHER	AN	BIKE	TR
PAGODA	BU	STATION	BU
TUBA	MI	SHED	BU
OAK	TE	HOUSE	BU
WILLOW	TE	GARAGE	BU
MINE	BU	PUB	BU
MANTIS	IN	CASTLE	BU
LUTE	MI	CHALET	BU
OWL	BI	CITADEL	BU
KRYPTON	EL	BEER	BE
TUNGSTEN	EL	ALE	BE
VIENNA	CC	BRANDY	BE
ZAMBIA	WC	GIN	BE
STAR	AL	LEMONADE	BE
MARS	AL	BELT	CL
NEPTUNE	AL	TIGHTS	CL
ASTEROID	AL	PICCOLO	MI
GALAXY	AL	EAGLE	BI
CAP	CL	HAWK	BI
BOOT	CL	PELICAN	BI
SLIPPER	CL	DOVE	BI
SKIRT	CL	TROMBONE	MI

ASH	TE	PUFFIN	BI
MUSCLE	HA	GNU	AN
MARTIN	BI	SNAIL	AN
MANDOLIN	MI	JAY	BI
LORIS	AN	XENON	EL
LARK	BI	HORN	MI
OSPREY	BI	COYPU	AN
URN	KU	SEPIA	CO
LEU	CU	MAGPIE	BI
JAW	HA	RUDD	FI
LODGE	BU		

CLASSIFICATION	CLUE
AN	ANIMAL
ET	ENGLISH TOWN
FI	FISH
EC	ENGLISH COUNTY
TR	A MODE OF TRANSPORT
MI	A MUSICAL INSTRUMENT
FL	A FLOWER
FR	A FRUIT
CO	A COLOUR
IN	AN INSECT
WC	A COUNTRY
SP	A SPORT
CT	A CONTINENT
CU	A CURRENCY
GC	A GREAT COMPOSER
VE	A VEGETABLE
CC	A CAPITAL CITY
EL	AN ELEMENT
AL	AN ASTRONOMICAL BODY
CL	AN ITEM OF CLOTHING
KU	A KITCHEN UTENSIL

HA	A PART OF THE BODY
BU	A BUILDING
BE	A BEVERAGE
BI	A BIRD
TE	A TREE

Words with clues

USE	APPLY
OGLE	STARE
CAT	FELINE
VEX	IRRITATE
PUZZLE	TEASER
OBIT	DATE OF DEATH
FIGS	EXOTIC FRUITS
RAT	RODENT
AFTER	BEHIND
EARL	ARISTOCRAT
ABIDE	WAIT FOR
DOVETAIL	A JOINT
NOD	BECK
FAME	GLORY
HOLINESS	BLESSEDNESS
RAKE	COLLECT
MUD	DIRT
GASP	PANT
CONFOUND	AMAZE
BET	STAKE
POVERTY	PAUPERISM
TOP	SPINNING – – –
STOVE	COOKER
FREEZER	ICE BOX
EGAD	BY GOD
UGLY	HORRIBLE
DOG	CANINE
ASIR	REGION OF ARABIA
JUVENILE	PEURILE
OBI	JAPANESE GARMENT
AFOOT	ASTIR
AFORE	IN FRONT OF
RADICES	ORIGINS
TRAVEL	JOURNEY
YALE	AMERICAN UNIVERSITY

YEARN	PINE
FACT	TRUTH
FAIN	EAGER
JEER	DERIDE
RAID	ATTACK
GANG	BAND
CONFIRM	ASSURE
BEHOLD	CONSIDER
RELY	DEPEND
LID	PAN TOP
COOKER	STOVE
FRIDGE	COOLER
SINK	WASHING BOWL
DRYER	TUMBLE – – – – –
SKEWER	STAKE
EAR	HUMAN SOUND RECEPTOR
SHIN	LEGBONE
SCISSORS	CUTTING IMPLEMENT
KIDNEY	A VITAL ORGAN
LIVER	A VITAL ORGAN
ARTERIES	BLOOD CANALS
ULNA	ARM BONE
POLISH	SHINE
ICE	COLD WATER
ELIXIR	TONIC
EXIT	WAY OUT
EASE	COMFORT
ACID	NOT ALKALINE
WET	NOT DRY
EJECT	THROW OUT
KEY	OPENER
QUIET	CALM
NAKED	BARE
NAVAL	MARINE
SMALL	TINY
TITTUP	MOVE
ADIT	APPROACH
FAIL	CEASE
DESPAIR	LOSE HOPE
SAY	DECLARE
ARCH	CHIEF
ESPY	DETECT
GAUNT	ANGULAR
KIN	RELATION
WRY	ASKEW

VERTICAL	UPRIGHT
CLOWN	BUFFOON
SUM	TOTAL
STRIP	UNDRESS
WOO	COURT
UNISON	HARMONY
KING	MONARCH
KIND	GENUS
ANOINT	SANCTIFY
APOTHEGM	DICTUM
ASIDE	APART
INANE	FUTILE
ADEPT	ACCOMPLISHED
SYZYGY	POINT OF CONJUNCTION
BEMUSE	STUPEFY
INNER	INTERIOR
QUEEN	MONARCH
SUGAR	SWEET
RAMIFY	DIVIDE
PUNGENT	NASTY SMELL
FEE	BILL
HORDE	MULTITUDE
DAINTY	CHARMING
MOTTO	INSCRIPTION
SHARP	POINTED
ZOO	ANIMAL PARK
ROC	LEGENDARY BIRD
ADVISE	RECOMMEND
OFFAL	LIVER
VASES	JUGS
SINGE	LIGHTLY BURN
CUPBOARD	LARDER
EYE	HUMAN LIGHT RECEPTOR
HAIR	FOUND ON HUMAN HEAD
LAB	SCIENCE WORKSHOP
BRUSH	– – – – – AND COMB
HEART	A VITAL ORGAN
BAT	FLYING MAMMAL
POINT	DIRECT
PLUMP	FAT
JIVE	DANCE
SCAR	MARK
EXOTIC	STRANGE
INK	WRITING FLUID
ALSO	AS WELL

UNIT	ONE
WAR	BATTLE
FORD	CROSSING
QUITE	TOTALLY
SABRE	SWORD
NASTY	DIRTY
SMART	KEEN
IDOL	DEITY
HICCUP	NOT TO PLAN
GOYA	ARTIST
DESPISE	SCORN
CONFLICT	BATTLE
APPROACH	ADVANCE
ACRE	LAND MEASURE
GAP	BREACH
GAZE	GAPE
VICINITY	NEARNESS
DOLE	ALLOCATE
ABUNDANT	AMPLE
HUMID	MOIST
HOCUS	STUPEFY
UNEASY	DISTURBED
SIXTY	FEMALE RETIREMENT AGE
KNIT	BIND
ANNALS	ARCHIVES
ANSWER	SOLUTION
SOPPING	SOAKED
INAPT	UNFIT
ADAPT	EVOLVE
ORBIT	COURSE
CORYZA	CATARRH
ALMOND	A NUT
SADLY	SORROWFULLY
PLASTIC	MATERIAL
PUNSTER	A MAKER OF PUNS
IGNORE	NEGLECT
GLEAN	COLLECT
FELON	CONVICT
CUSTOM	CONVENTION
ACRID	PUNGENT
SISAL	STRONG FABRIC
START	COMMENCE
ZIP	FASTENER
ORC	MYTHICAL CREATURE
MINGLE	BLEND

OFFER	BLEND
JESUS	SAVIOUR
SIREN	AIR-RAID WARNING
SISSY	EFFEMINATE BOY
ZOMBI	WALKING DEAD
JEWEL	PRECIOUS STONE
KERRY	COUNTY IN IRELAND
LOGIC	CHAIN OF REASONING
SIGHT	FACULTY OF VISION
VALUE	WORTH
ALLAH	MOSLEM GOD
PACIFY	AMELIORATE
SLY	ARTFUL
VIE	CONTEST
FATUOUS	INFATUATED
IMMORAL	DEBAUCHED
MILIEU	ENVIRONMENT
CHOP	MINCE
MOIETY	PORTION
MONASTIC	RECLUSE
NAG	HARASS
NEUTRAL	IMPARTIAL
OGRE	SPECTRE
ORAL	VOCAL
PEER	ASSOCIATE
POWERFUL	FORCEFUL
SPASM	PAROXYSM
SURGE	SWELL
TACIT	IMPLICIT
TARNISH	DISCOLOUR
THICKET	WOODLAND
TITTLE	PARTICLE
TRUISM	AXIOM
UNCANNY	PRETERNATURAL
VENT	EXPRESS
EXCITE	AROUSE
FACTION	PARTY
DIVINE	FATHOM
FLOW	GLIDE
FRACAS	AFFRAY
FRET	ERODE
FUSE	MERGE
GAGE	PLEDGE
PADDY	AN IRISHMAN
REFUSAL	REJECTION

JETTY	PIER
JACOB	SON OF ISAAC
JUMPY	NERVOUS
POSTMAN	DELIVERS LETTERS
BEDIM	DARKEN
SATIN	FINE CLOTH
ERROR	MISTAKE
EXTOL	APPLAUD
HARDY	STOUT-HEARTED
RAVINE	GORGE
PEEVISH	CHILDISH
CRY	LAMENT
GAMBOL	FROLIC
EXTEND	LENGTHEN
PRY	SEARCH
PYGMY	DWARF
QUELL	CONQUER
QUIVER	ARROW HOLDER
RACE	CHASE
QUIXOTIC	FANCIFUL
LAD	YOUNG MAN
SECTS	FACTIONS
SCREW	TIGHTEN
HOURI	VOLUPTUOUS FEMALE
KETCH	COASTAL VESSEL
KHAKI	ARMY FABRIC
MAORI	NATIVE OF NEW ZEALAND
WEDGE	TAPERED PIECE OF WOOD
SABLE	SMALL ARCTIC ANIMAL
ALLOY	A MIXTURE OF METALS
PARADOX	MYSTERY
UNABLE	INCAPABLE
DISCIPLE	CATECHUMEN
IMPEL	INCITE
BESTOW	IMPART
METE	ALLOT
MITIGATE	ASSUAGE
APPEASE	MOLLIFY
MYTH	LEGEND
NAUSEA	REPUGNANCE
NUNCIO	ENVOY
OPIATE	ANODYNE
PALTRY	CONTEMPTIBLE
PERPLEX	BESET
SNEER	DERIDE

SUE	PETITION
SYNOPSIS	OUTLINE
TAMPER	ALTER
TETHER	MANACLE
THRIFT	ECONOMY
TROUBLE	INCONVENIENCE
TYRO	NOVICE
UNCOUTH	BOORISH
EVIL	CORRUPT
FACILE	COURTEOUS
FALLACY	DECEIT
FLEET	NAVY
FOP	BEAU
FREE	LIBERATE
FRIEND	COMPANION
FUN	MIRTH
SIDES	EDGES
PADRE	PRIEST
GHASTLY	HORRIBLE
QUA	IN THE CAPACITY OF
JUMBO	ELEPHANT
FIREMAN	FIGHTS FIRES
XYLEM	PART OF PLANT
BADGE	EMBLEM
BIPED	TWO-LEGGED
ASSAY	EXAMINE
FABLE	PARABLE
SKILL	ABILITY
LIP	PART OF MOUTH
PITY	COMPASSION
PINE	YEARN
PLOD	TRUDGE
PROSPER	FLOURISH
STEAL	PILFER
QUALM	SCRUPLE
QUIT	ABANDON
QUIZ	CONUNDRUM
RAISE	ELEVATE
RARITY	SCARCITY
SALT	SALINE
ELBOW	ARM JOINT
HABIT	TENDENCY
SULKY	CHURLISH
STRAP	BELT
SAKER	LARGE FALCON

SALARY	INCOME
SAUCY	FLIPPANT
TOY	PLAYTHING
TRAIT	CHARACTERISTIC
UMPIRE	JUDGE
VALLEY	DALE
VAST	BOUNDLESS
WARN	CAUTION
WARP	BEND
WINNOW	SEPARATE
WONTED	CUSTOMARY
ABRADE	ERRODE
ABUT	BORDER
BELLOW	CLAMOUR
BRIM	EDGE
CHAGRIN	MORTIFICATION
CAUSTIC	CORROSIVE
CLOTHE	APPAREL
COERCE	SUBDUE
DETAIL	DESCRIPTION
DIFFUSE	SPREAD
SUITE	ESCORT
ROOMY	EXTENSIVE
ABBEY	PRIORY
LAIR	HOME OF BEAST
SATIRE	IRONY
TOTAL	WHOLE
TRACE	OUTLINE
ULCER	BOIL
VAIN	TRIVIAL
VAPOUR	FUME
VET	ANIMAL DOCTOR
WARRANT	DECLARE
WILY	CUNNING
WANT	REQUIRE
ABLUTION	CLEANSING
ACME	PINNACLE
ABYSS	BOTTOMLESS PIT
BEWAIL	EXPRESS SORROW
BULWARK	GUARD
CEDE	ABDICATE
CLONE	EXACT COPY
COMA	UNCONSCIOUS STATE
DESPOT	AUTOCRAT
DICTATOR	DESPOT

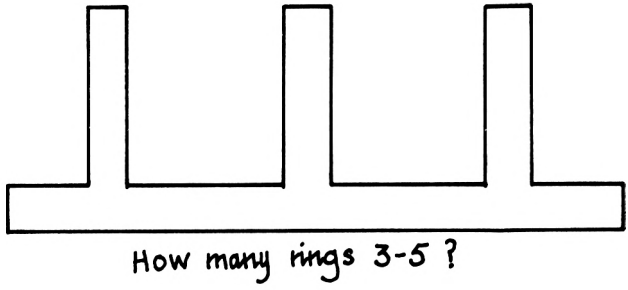
6. Towers of Hanoi

Towers of Hanoi is a graphical puzzle which requires a large amount of concentration and logical thinking. The object of the game is to transport three, four or five rings from tower 1 to tower 3 (see below). You will see an animated portrayal as each ring is transported across the screen.

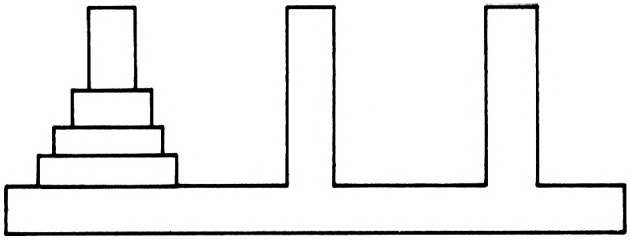
A clock runs continuously throughout the puzzle. It displays the time you are taking to solve it. When you have finished the puzzle, the three lowest times (one for three rings, one for four and one for five) are recorded on to cassette.

Playing instructions

When the program is executed and the data containing the record times has been loaded from cassette, you will be greeted by the following picture:



If you respond by typing 3 then three rings will appear at tower 1 (see below). Typing 4 or 5 will cause the corresponding number of rings to appear.



The record time for the amount of rings chosen will then appear, and you will be asked if you wish to see the solution.

When moving a ring, first type the number corresponding to its starting position (1-3) and then its destination, but the rule of the puzzle is that no ring may be placed on any ring of smaller size, and should you attempt to do this the computer will display an error message. The puzzle is solved when all the pieces have been transferred to tower 3 in the same pattern as they started on tower 1, i.e. smallest on the top and largest on the bottom.

Note that the smallest number of moves required to complete the puzzle can be calculated mathematically as:

minimum number of moves = 2^r-1

where *r* is the number of rings.

The program

Towers of Hanoi requires a small file on cassette in which to store the record times, and this file must be created before the program is executed. This is achieved by following the next two steps carefully.

Step 1: Type in the main program and save it on to one side of a blank cassette. *Do not rewind the tape.*

Step 2: Type in the short file-creation program listed at the end of this chapter and execute it immediately. There is no need to save the program since it will automatically create the file and save it on to cassette, and will never be needed again.

Towers of Hanoi is an animated graphical puzzle and it relies heavily on the graphics facilities of the Amstrad. Special care must therefore be taken when entering lines 810-1530, as these lines are concerned with the movement of the rings.

Throughout the puzzle a timer is in operation, and this is controlled by an EVERY interrupt. It is in the nature of such commands to interrupt at the most unwanted times, so to stop the graphics routines from being interrupted, the commands DI (Disable Interrupts) and EI (Enable Interrupts) have been incorporated in this program.

Flowchart

The flowchart on p.109 represents the general operation of the program.

Variables

The following table represents the major variables used in the program and should help you to understand how the program operates.

RE$()	Array holding the record times
RES$()	Array holding the record holders' names
B(x,y)	Contents of the three towers
MI	Minutes passed
SEC	Seconds passed
RING	Number of rings
SOL	Indicates a computer solution in progress
Q	Tower to move from
W	Destination tower

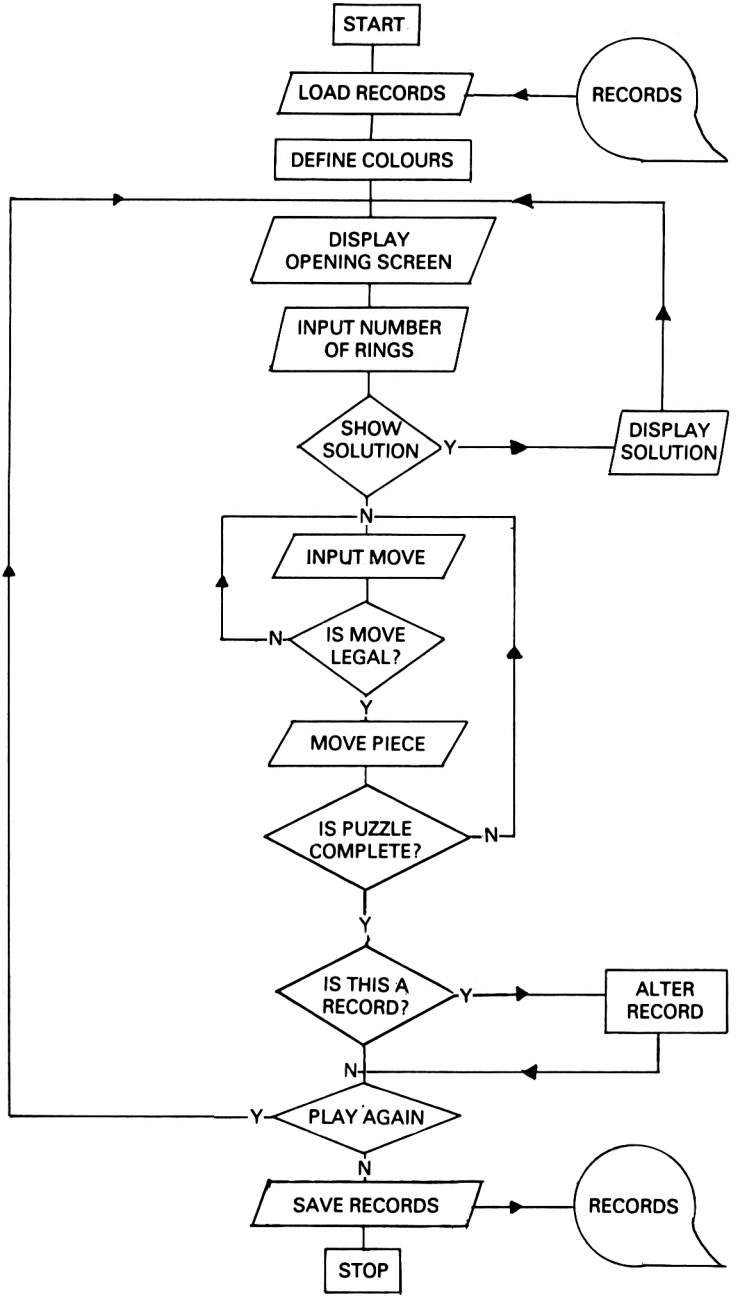
Comments on the program

Lines 10-20
This routine transfers the record times from the cassette to the computer.

Lines 30-100
The main variables and colours to be used during the program are initialised.

Lines 110-160
This section displays the opening screen.

Lines 170-330
In this routine the computer will ask you how many rings you require and it will adjust the variable B(x,y) accordingly, printing the rings on the screen.


START
LOAD RECORDS
RECORDS
DEFINE COLOURS
DISPLAY OPENING SCREEN
INPUT NUMBER OF RINGS
SHOW SOLUTION
Y
DISPLAY SOLUTION
N
INPUT MOVE
N
IS MOVE LEGAL?
Y
MOVE PIECE
IS PUZZLE COMPLETE?
N
Y
IS THIS A RECORD?
Y
ALTER RECORD
N
Y
PLAY AGAIN
N
SAVE RECORDS
RECORDS
STOP

Lines 340-420
This routine will ask if you wish to see the computer perform the solution of the puzzle, and if you require this facility, the variable SOL will be given the value 1.

Lines 430-440
These two lines initialise the interrupt for the timer and display the word TIME on the screen.

Lines 450-700
This routine inputs the player's move from the keyboard and checks to ensure that it is legal. If the computer is demonstrating the solution then the moves are read from data.

Lines 710-800
This sequence of instructions calls one of the seven routines that will move the ring.

Lines 810-1530
These lines contain the seven separate routines which perform similar tasks:

Lines 810-950	Move a ring up
Lines 960-1050	Move a ring right one place
Lines 1060-1150	Move a ring left one place
Lines 1160-1330	Move a ring down
Lines 1340-1430	Move a ring right two places
Lines 1440-1530	Move a ring left two places

Lines 1540-1590
The data required by the computer to demonstrate the solution of the puzzle.

Lines 1600-1760
Tower 3 is checked to see if it holds all the rings in the correct order. If this is the case, the puzzle has been completed. If the time taken to complete the puzzle is less than the record time then this is amended and you are asked if you wish to play again. If you do not, the record times will be recorded on to cassette.

Lines 1770-1830
This is the timer interrupt routine in which the time is increased by one second each time the routine is called. If the timer reaches ten minutes you will be told that you are out of time and the game will finish.

The listings

The main program is given first and then the file creation program. They should be entered very carefully and saved as instructed in the text of this chapter.

The game should be entered and played in the caps lock mode. All spaces within the text should be entered as they appear.

MAIN PROGRAM

```
10 INK 0,0:PAPER 0:INK 1,24:PEN 1:BORDER 0:MODE 1:
LOCATE 1,11:PRINT" ENTER DATA CASSETTE TO LOAD REC
ORDS.":PRINT
20 OPENIN "":FOR I=3 TO 5:INPUT#9,RE$(I),RES$(I):N
EXT I:CLOSEIN
30 DIM B(3,5)
40 INK 0,0:INK 1,24:INK 2,13
50 INK 3,15
60 PAPER 0:PEN 1:BORDER 0
70 LET MI=0:LET SEC=0
80 MODE 1
90 LET A=1
100 PEN 2
110 PEN 1:LOCATE 13,2:PRINT"TOWERS OF HANOI":PRINT
:PRINT"THE OBJECT IS TO MOVE THE RINGS TO NO.3"
120 FOR J=20 TO 21
130 PEN 2
140 FOR I=2 TO 38:LOCATE I,J:PRINT CHR$(143):NEXT
I,J
150 FOR I=8 TO 38 STEP 12:FOR J=14 TO 19
160 LOCATE I,J:PRINT CHR$(143):NEXT J:PEN 1:LOCATE
 I,21:PRINT RIGHT$(STR$(A),1):A=A+1:PEN 2:NEXT I
170 PEN 1
180 LOCATE 1,25:INPUT"HOW MANY RINGS (3-5) ";RING$
190 IF VAL(RING$)<3 OR VAL(RING$)>5 THEN GOTO 180
ELSE LET RING=VAL(RING$)
200 FOR I=1 TO 3:FOR J=1 TO 5:LET B(I,J)=0:NEXT J,
I
210 LOCATE 11,23:PRINT"RECORD ";RE$(RING);" BY ";R
ES$(RING)
220 LOCATE 1,25:PRINT SPC(38)
230 LET A=6-RING
240 PEN 3
250 FOR I=3 TO 11 STEP 2
260 LET B(1,A)=I
270 LET A=A+1
280 IF A<6 THEN NEXT I
290 LET A=5
300 FOR J=19 TO 20-RING STEP-1
```

```
310 LOCATE ((B(1,A)-1)/2)+(9-B(1,A)),J
320 FOR I=1 TO B(1,A):PRINT CHR$(143);:NEXT I
330 LET A=A-1:IF A>5-RING THEN NEXT J
340 PEN 1:LOCATE 1,25:PRINT"SEE THE SOLUTION ? (Y
OR N)"
350 LET Q$=INKEY$:IF Q$="" THEN GOTO 350
360 IF Q$="Y" THEN LET SOL=1
370 IF Q$="N" THEN LET SOL=0
380 IF Q$<>"Y" AND Q$<>"N" THEN GOTO 350
390 LOCATE 1,25:PRINT "
          "
400 IF RING=3 THEN RESTORE 1540
410 IF RING=4 THEN RESTORE 1560
420 IF RING=5 THEN RESTORE 1580
430 LOCATE 14,25:PEN 1:PRINT"TIME: 0:00"
440 EVERY 50 GOSUB 1770
450 FOR I=1 TO 20:LET Q$=INKEY$:NEXT I
460 DI:LOCATE 1,6:LOCATE 1,7:PRINT "        ":LOCAT
E 1,8:PRINT"        "
470 PEN 1:LOCATE 1,6:PRINT"FROM: ":LOCATE 6,6:EI
480 IF SOL=1 THEN READ Q:LET Q$=RIGHT$(STR$(Q),1):
GOTO 520
490 LET Q$=INKEY$:IF Q$="" THEN GOTO 490
500 IF Q$="Q" THEN GOTO 1700
510 IF VAL(Q$)<1 OR VAL(Q$)>3 THEN GOTO 470
520 DI:LOCATE 6,6:PRINT Q$:EI
530 LET Q=VAL(Q$)
540 FOR I=1 TO 5
550 IF B(Q,I)<>0 THEN GOTO 610
560 NEXT I
570 DI:LOCATE 1,9:PRINT"THERE ARE NO RINGS THERE."
:EI
580 FOR I=1 TO 1600:NEXT I
590 DI:LOCATE 1,9:PRINT SPC(25):EI
600 GOTO 450
610 DI:LOCATE 1,7:PRINT"  TO: ":LOCATE 6,7:EI
620 IF SOL=1 THEN READ P:LET Q$=RIGHT$(STR$(P),1):
GOTO 660
630 LET Q$=INKEY$:IF Q$="" THEN GOTO 610
640 IF Q$="Q" THEN GOTO 1770
650 IF VAL(Q$)<1 OR VAL(Q$)>3 OR VAL(Q$)=Q THEN GO
TO 610
660 DI:LOCATE 6,7:PRINT Q$:EI
670 LET W=VAL(Q$)
680 FOR D=1 TO 5
690 IF D<6 AND B(W,D)=0 THEN NEXT D:GOTO 710
700 IF B(W,D)<B(Q,I) THEN DI:LOCATE 1,8:PRINT"ILLE
GAL MOVE.":FOR K=1 TO 1600:NEXT K:LOCATE 1,7:PRINT
"     ":LOCATE 1,8:PRINT"              ":EI:GOTO 45
0
710 DI:GOSUB 810
```

```
720 IF W-Q=1 THEN GOSUB 960
730 IF W-Q=2 THEN GOSUB 1340
740 IF W-Q=-1 THEN GOSUB 1060
750 IF W-Q=-2 THEN GOSUB 1440
760 GOSUB 1160
770 EI
780 IF W=3 THEN GOSUB 1600
790 GOTO 450
800 END
810 REM **MOVE PIECE UP**
820 IF Q=1 THEN LET X=9-B(Q,I)
830 IF Q=2 THEN LET X=21-B(Q,I)
840 IF Q=3 THEN LET X=33-B(Q,I)
850 FOR I=1 TO 5
860 IF B(Q,I)<>0 THEN GOTO 880
870 NEXT I
880 FOR J=20-(7-I) TO 10 STEP -1
890 PEN 3
900 LOCATE ((B(Q,I)-1)/2)+X,J
910 FOR K=1 TO B(Q,I):PRINT CHR$(143);:NEXT K
920 LOCATE ((B(Q,I)-1)/2)+X,J+1:PRINT SPC(B(Q,I))
930 IF J>12 THEN PEN 2:LOCATE 8+(12*(Q-1)),J+1:PRI
NT CHR$(143):PEN 3
940 NEXT J
950 RETURN
960 REM **RIGHT ONE PLACE**
970 PEN 3
980 FOR J=((B(Q,I)-1)/2)+X TO ((B(Q,I)-1)/2)+X+11
990 IF J>5 THEN LET P=J-4 ELSE P=J-2
1000 LOCATE P,10:PRINT "       ":LOCATE J+1,10
1010 FOR K=1 TO B(Q,I):PRINT CHR$(143);:NEXT K
1020 FOR L=1 TO 10:NEXT L
1030 NEXT J
1040 LET X=J+(((B(Q,I)-1)/2)-2)
1050 RETURN
1060 REM **LEFT ONE PLACE**
1070 PEN 3
1080 FOR J=X TO ((B(Q,I)-1)/2)+X-12 STEP -1
1090 IF J>5 THEN LET P=J-4 ELSE P=J-2
1100 LOCATE J+B(Q,I)+1,10:PRINT "      ":LOCATE J+1
,10
1110 FOR K=1 TO B(Q,I):PRINT CHR$(143);:NEXT K
1120 FOR L=1 TO 10:NEXT L
1130 NEXT J
1140  LET X=J+(((B(Q,I)-1)/2)-1)
1150 RETURN
1160 REM **MOVE PIECE DOWN**
1170 IF W=1 THEN LET X=9-B(Q,I)
1180 IF W=2 THEN LET X=21-B(Q,I)
1190 IF W=3 THEN LET X=33-B(Q,I)
1200 LET X=((B(Q,I)-1)/2)+X
```

```
1210 FOR S=1 TO 5
1220 IF B(W,S)<>0 THEN GOTO 1240
1230 NEXT S
1240 FOR J=10 TO 20-(7-S)
1250 LOCATE X,10:PRINT SPC(15)
1260 PEN 3
1270 LOCATE X,J
1280 FOR K=1 TO B(Q,I):PRINT CHR$(143);:NEXT K
1290 LOCATE X,J-1:PRINT SPC(B(Q,I))
1300 IF J>14 THEN PEN 2:LOCATE 8+(12*(W-1)),J-1:PR
INT CHR$(143):PEN 3
1310 NEXT J
1320 LET B(W,S-1)=B(Q,I):LET B(Q,I)=0
1330 RETURN
1340 REM **TWO PLACES RIGHT**
1350 PEN 3
1360 FOR J=12-B(Q,I) TO 33-B(Q,I)-(5-B(Q,I))+2
1370 IF J>5 THEN LET P=J-4 ELSE P=J+2
1380 LOCATE P,10:PRINT "      ":LOCATE J+1,10
1390 FOR K=1 TO B(Q,I):PRINT CHR$(143);:NEXT K
1400 FOR L=1 TO 10:NEXT L
1410 NEXT J
1420 LET X=J-1
1430 RETURN
1440 REM **TWO PLACES LEFT**
1450 PEN 3
1460 FOR J=33-B(Q,I)-(5-B(Q,I)) TO 9-B(Q,I)+((B(Q,
I)-1)/2)-1 STEP -1
1470 IF J>5 THEN LET P=J-4 ELSE P=J-2
1480 LOCATE J+B(Q,I)+1,10:PRINT "      ":LOCATE J+1
,10
1490 FOR K=1 TO B(Q,I):PRINT CHR$(143);:NEXT K
1500 FOR L=1 TO 10:NEXT L
1510 NEXT J
1520 LET X=J+1
1530 RETURN
1540 REM **RING=3**
1550 DATA 1,3,1,2,3,2,1,3,2,1,2,3,1,3
1560 REM
1570 DATA 1,2,1,3,2,3,1,2,3,1,3,2,1,2,1,3,2,3,2,1,
3,1,2,3,1,2,1,3,2,3
1580 REM
1590 DATA 1,3,1,2,3,2,1,3,2,1,2,3,1,3,1,2,3,2,3,1,
2,1,3,2,1,3,1,2,3,2,1,3,2,1,2,3,1,3,2,1,3,2,3,1,2,
1,2,3,1,3,1,2,3,2,1,3,2,1,2,3,1,3
1600 REM **CHECK FOR COMPLETE PUZZLE**
1610 FOR D=(5-RING)+1 TO 5
1620 IF B(3,D)=0 THEN RETURN
1630 LET A(D)=B(3,D)
1640 IF A(D)>A(D-1)THEN NEXT D ELSE RETURN
1650 LET D=REMAIN(0):LOCATE 1,6:PEN 1:PRINT"PUZZLE
```

```
 COMPLETE"
1660 IF (MI*60)+SEC>((VAL(LEFT$(RE$(RING),1)))*60)
+(VAL(RIGHT$(RE$(RING),2))) THEN GOTO 1700
1670 IF SOL=1 THEN GOTO 1700
1680 LET RE$(RING)=RIGHT$(STR$(MI),1)+":"+RIGHT$(S
TR$(SEC),2)
1690 LOCATE 1,7:PRINT"YOU HAVE THE RECORD.":INPUT"
ENTER YOUR NAME.(3 LETTERS) ";RES$(RING):LET RES$(
RING)=LEFT$(RES$(RING),3)
1700 LET D=REMAIN(0):LOCATE 1,6:PRINT"PUZZLE COMPL
ETE":PRINT"                              ":LOCATE 1,8:P
RINT"PLAY AGAIN ? (Y OR N)                "
1710 LET Q$=INKEY$:IF Q$="" THEN GOTO 1710
1720 IF Q$="Y" THEN GOTO 70
1730 IF Q$="N" THEN GOTO 1740 ELSE GOTO 1710
1740 MODE 1:LOCATE 1,11:PRINT" ENTER DATA CASSETTE
 TO SAVE RECORDS.":PRINT
1750 OPENOUT "RECORDS":FOR I=3 TO 5:PRINT#9,RE$(I)
:PRINT#9,RES$(I):NEXT I:CLOSEOUT
1760 END
1770 REM **TIMER**
1780 PEN 1
1790 LET SEC=SEC+1:IF SEC=60 THEN LET SEC=0:LET MI
=MI+1
1800 LOCATE 20,25:PRINT USING"#";MI;:PRINT ":";:PR
INT USING"##";SEC
1810 IF SEC<10 THEN LOCATE 22,25:PRINT"0"
1820 IF SEC=0 AND MI=10 THEN LOCATE 1,6:PRINT"OUT
OF TIME.":GOTO 1700
1830 RETURN
```

FILE CREATION PROGRAM

```
10 MODE 1:LOCATE 1,11:PRINT" ENTER DATA CASSETTE T
O SAVE RECORDS.":PRINT
20 OPENOUT "RECORDS"
30 FOR I=1 TO 6:READ A$:PRINT #9,A$:NEXT I
40 CLOSEOUT
50 DATA "0:44","SJL","1:30","SJL","2:50","SJL"
```

Part Two

Games

7. Othello

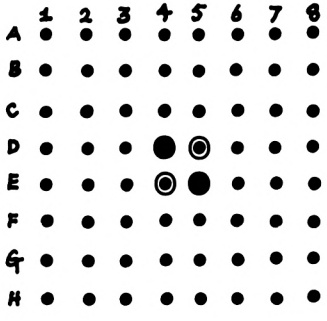
If you have played the game Othello or Reversi before, now is your chance to take on the ultimate challenge and attempt to defeat the computer.

The game is played on an eight by eight board with sixty-four pieces of two colours (in our case red and blue) which are placed alternately on the board. The object of the game is to fill the board with as many of your pieces as possible by capturing your opponent's pieces.

Playing instructions

When the game is executed you will be asked to input the number of players (one or two).

In the two-player mode, the computer will provide you with the board and act as the adjudicator, but if one player is accepted then you must prepare yourself for a struggle with the mighty Amstrad. When the number of players and their names have been entered you will be required to decide who is to have the first move, after which the board will be displayed on the screen as shown below.



Each player then takes his turn to place a counter on the board if a legal move is available. A legal move is defined as one which will capture one or more of the opponent's pieces by trapping them between two of your own. The counter is then placed on the board and all captured pieces are turned over to change their colour to that of your own counters.

All moves are entered by using the board co-ordinate system with the letters or rows first (e.g. A4, B3, etc.). If at any time during the game you are unable to make a legal move then you must pass by typing P and it automatically becomes your opponent's move.

The game is over when either the board contains counters of one colour only, or when the board is completely full, at which time the winner is declared to be the player with the majority of pieces on display.

The program

The program operates by using a system of weighting in which every possible move is considered and a weight calculated. This weight depends on several factors, including the number of pieces captured and the computer's overall position on the board. The computer plays a rather defensive game, but by considering the flowchart, table of variables and comment it should be a simple matter to adjust the weighting section and therefore experiment with your own strategy.

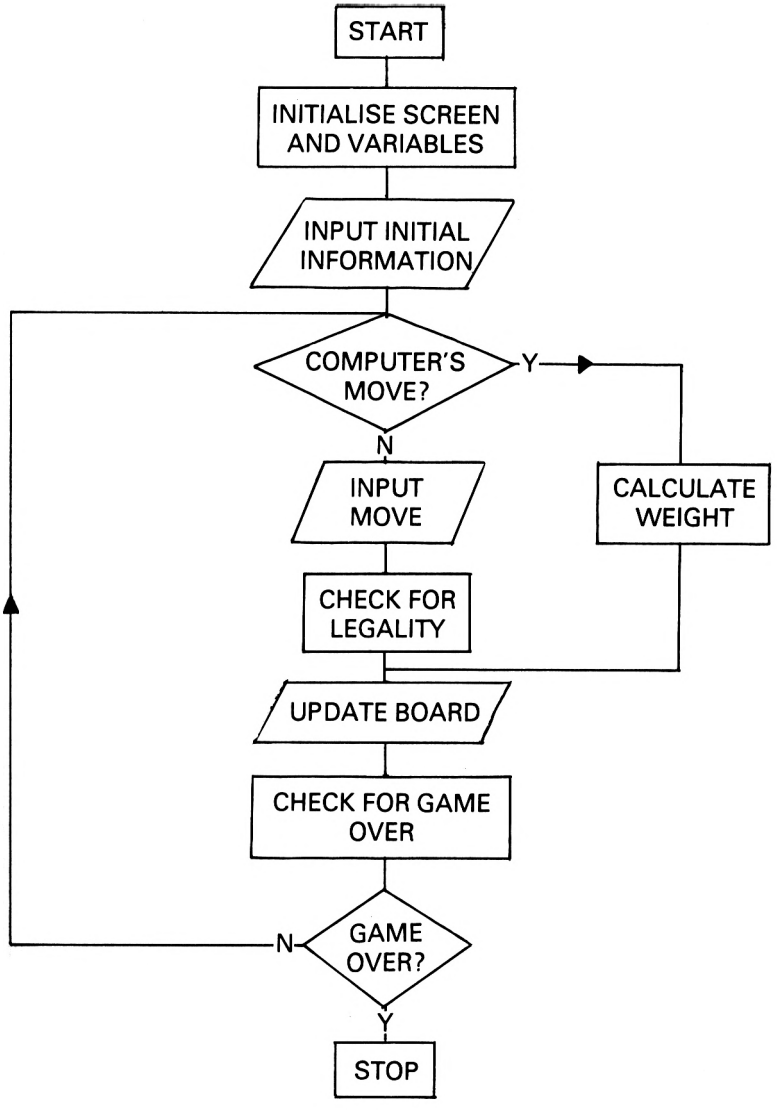
Flowchart

The flowchart (opposite) represents the general operation of the program if the one-player game is selected.

Variables

The following table represents the major variables used in the program and should be of assistance if you wish to adjust the game to play a different strategy.

A(x,y)	Array storing board
W(x,y)	Weights for moves
MOV	Move number
C	Used to indicate an illegal move


START
INITIALISE SCREEN AND VARIABLES
INPUT INITIAL INFORMATION
COMPUTER'S MOVE?
Y
N
INPUT MOVE
CALCULATE WEIGHT
CHECK FOR LEGALITY
UPDATE BOARD
CHECK FOR GAME OVER
GAME OVER?
N
Y
STOP

N$	Name of player two or computer
M$	Name of player one
JJ	White counter
QQ	Red counter

Comments on the program

This program contains two very large routines which control the legality checks and the evaluation of the best possible move for the computer, with the remainder of the program designed around these two sections.

Lines 10-230
These opening lines initialise the main variables and dimension the arrays used throughout the program.

Lines 240-520
The routine to control the general flow of the game, printing the appropriate messages in the left-hand window.

Lines 530-620
A section of code used to update the board on the screen.

Lines 630-740
A routine used by the legality check section to display an appropriate message on the screen.

Lines 750-1820
One of the two major routines which use the principles of artificial intelligence to check on the legality of a move made by the player. This is achieved by considering the updated board position and using the pattern recognition technique to search for legal combinations.

Lines 1830-1960
A routine called when there is no legal move for the computer. A message is displayed and it is then the player's turn.

Lines 1970-2160
The control loop used to calculate the best possible move for the computer.

Lines 2170-2210
A short routine to print the co-ordinates of the computer's move in the appropriate window.

Lines 2220-3310
The second major routine which considers all the possible legal moves for the computer and evaluates the best one available.

Lines 3320-3350
Time delays.

Lines 3360-3500
Initial routine used to display the board at the start of the game.

Lines 3510-3690
A short subroutine used at the end of the game to identify and declare the winning player.

Lines 3700-3930
This routine is used at the beginning of the program to collect the initial criteria for the game and implement the drawing of the board.

The listing

It is most important that special care is taken when entering the program as any errors could result in illegal moves being made by the computer. When entered the program should be saved on to cassette as outlined in the Introduction.

The game should be entered and played in the caps lock mode. All spaces within the text should be entered as they appear.

```
10 LET K=0
20 LET MOV=0
30 DIM A(10,10)
40 LET A(5,5)=-1
50 LET A(6,6)=-1
60 LET A(6,5)=1
70 LET A(5,6)=1
80 LET C=0
90 DIM W(10,10)
100 LET A$=""
110 FOR I=1 TO 10
120 LET W(I,1)=-8
130 LET W(I,10)=-8
```

```
140 LET W(1,I)=-8
150 LET W(10,I)=-8
160 NEXT I
170 LET QQ=0
180 LET JJ=0
190 LET EN=0
200 INK 1,24
210 INK 2,6
220 INK 0,0
230 INK 3,26
240 PEN 1:PAPER 0:BORDER 0:MODE 1
250 IF K=0 THEN GOSUB 3700
260 GOSUB 530
270 LET MOV=MOV+1
280 IF EN>1 THEN GOTO 3510
290 IF MOV/2<>INT(MOV/2) THEN GOTO 320
300 LOCATE#2,3,4:PEN#2,2:PRINT#2,"RED  "
310 GOTO 330
320 LOCATE#2,3,4:PEN#2,0:PRINT#2,"WHITE"
330 LOCATE#2,7,6:PRINT#2,"  "
340 IF C<>0 THEN PEN#2,0:LOCATE#2,8,9:PRINT#2,A$
350 LET C=0
360 IF MOV/2=INT(MOV/2) AND PL=1 THEN GOTO 1830
370 FOR I=1 TO 20:LET D$=INKEY$:NEXT I:LET A$=""
380 LET Q$=INKEY$:IF Q$="" THEN GOTO 380
390 LET A$=A$+Q$:LOCATE#2,6+LEN(A$),6:PRINT#2,Q$:I
F LEN(A$)=2 THEN GOTO 400 ELSE GOTO 380
400 IF A$="PP" THEN GOTO 1890
410 LET I=ASC(LEFT$(A$,1))-64
420 LET J=VAL(RIGHT$(A$,1))
430 IF J<1 OR J>8 OR I<1 OR I>8 THEN GOTO 660
440 LET I=I+1
450 LET J=J+1
460 LOCATE#2,7,6:PEN#2,0:PRINT#2,A$;" "
470 IF MOV/2=INT(MOV/2) AND PL=1 THEN GOTO 1830
480 GOTO 630
490 LET A(I,J)=-1
500 IF MOV/2=INT(MOV/2) THEN LET A(I,J)=1
510 GOSUB 530
520 GOTO 270
530 REM ***
540 PAPER 0
550 FOR F=4 TO 18 STEP 2
560 FOR G=15 TO 29 STEP 2
570 IF A(F/2,(G-11)/2)=1 THEN LOCATE G+1,F+1:PEN 2
:PRINT CHR$(231)
580 IF A(F/2,(G-11)/2)=-1 THEN LOCATE G+1,F+1:PEN
3:PRINT CHR$(231)
590 IF A(F/2,(G-11)/2)=0 THEN LOCATE G+1,F+1:PEN 1
:PRINT CHR$(144)
600 NEXT G
```

```
610 NEXT F
620 RETURN
630 LET MX=-2
640 IF A(I,J)=0 THEN GOTO 750
650 REM
660 FOR F=1 TO 3
670 LOCATE 1,1:PAPER 2:PEN 1:PRINT"ILLEGAL MOVE"
680 FOR O=0 TO 10
690 NEXT O
700 LOCATE 1,1:PAPER 2:PRINT"              "
710 FOR O=0 TO 10
720 NEXT O
730 NEXT F
740 GOTO 280
750 REM
760 LET AA=-1
770 IF MOV/2=INT(MOV/2) THEN LET BB=-1
780 IF MOV/2=INT(MOV/2) THEN LET AA=1
790 FOR L=J+1 TO 10
800 IF A(I,L)=0 THEN GOTO 880
810 IF A(I,L)=AA THEN GOTO 830
820 NEXT L
830 IF L=J+1 THEN GOTO 880
840 FOR M=J+1 TO L-1
850 LET A(I,M)=AA
860 NEXT M
870 LET C=C+1
880 FOR L=J-1 TO 1 STEP-1
890 IF A(I,L)=0 THEN GOTO 970
900 IF A(I,L)=AA THEN GOTO 920
910 NEXT L
920 IF L=J-1 THEN GOTO 970
930 FOR M=J-1 TO L+1 STEP-1
940 LET A(I,M)=AA
950 NEXT M
960 LET C=C+1
970 FOR L=I+1 TO 10
980 IF A(L,J)=0 THEN GOTO 1060
990 IF A(L,J)=AA THEN GOTO 1010
1000 NEXT L
1010 IF L=I+1 THEN GOTO 1060
1020 FOR M=I+1 TO L-1
1030 LET A(M,J)=AA
1040 NEXT M
1050 LET C=C+13
1060 FOR L=I-1 TO 1 STEP-1
1070 IF A(L,J)=0 THEN GOTO 1150
1080 IF A(L,J)=AA THEN GOTO 1100
1090 NEXT L
1100 IF L=I-1 THEN GOTO 1150
1110 FOR M=I-1 TO L+1 STEP -1
```

```
1120 LET A(M,J)=AA
1130 NEXT M
1140 LET C=C+1
1150 REM ***DIAGONALS***
1160 LET Z=1
1170 LET M=J+Z
1180 LET EN=EN+1
1190 LET L=I-Z
1200 IF J<10 AND I<10 THEN LET EN=0
1210 IF L=1 OR M=10 THEN GOTO 1340
1220 IF J=10 OR I=10 THEN GOTO 1890
1230 IF A(L,M)=0 THEN GOTO 1340
1240 IF A(L,M)=AA THEN GOTO 1270
1250 LET Z=Z+1
1260 GOTO 1170
1270 IF Z=1 THEN GOTO 1340
1280 FOR V=1 TO Z-1
1290 LET L=I-V
1300 LET M=J+V
1310 LET A(L,M)=AA
1320 NEXT V
1330 LET C=C+1
1340 REM ***NEXT DIAG***
1350 LET Z=1
1360 LET L=I+Z
1370 LET M=J-Z
1380 IF L=10 OR M=1 THEN GOTO 1500
1390 IF A(L,M)=0 THEN GOTO 1500
1400 IF A(L,M)=AA THEN GOTO 1430
1410 LET Z=Z+1
1420 GOTO 1360
1430 IF Z=1 THEN GOTO 1500
1440 FOR V=1 TO Z-1
1450 LET L=I+V
1460 LET M=J-V
1470 LET A(L,M)=AA
1480 NEXT V
1490 LET C=C+1
1500 LET Z=1
1510 LET L=I+Z
1520 LET M=J+Z
1530 IF L=10 OR M=10 THEN GOTO 1650
1540 IF A(L,M)=0 THEN GOTO 1650
1550 IF A(L,M)=AA THEN GOTO 1580
1560 LET Z=Z+1
1570 GOTO 1510
1580 IF Z=1 THEN GOTO 1650
1590 FOR V=1 TO Z-1
1600 LET L=I+V
1610 LET M=J+V
1620 LET A(L,M)=AA
```

```
1630 NEXT V
1640 LET C=C+1
1650 LET Z=1
1660 LET L=I-Z
1670 LET M=J-Z
1680 IF L=1 OR M=1 THEN GOTO 1800
1690 IF A(L,M)=0 THEN GOTO 1800
1700 IF A(L,M)=AA THEN GOTO 1730
1710 LET Z=Z+1
1720 GOTO 1660
1730 IF Z=1 THEN GOTO 1800
1740 FOR V=1 TO Z-1
1750 LET L=I-V
1760 LET M=J-V
1770 LET A(L,M)=AA
1780 NEXT V
1790 LET C=C+1
1800 REM ***END OF SEQUENCE***
1810 IF C=0 THEN GOTO 650
1820 GOTO 490
1830 LET MX=-6
1840 PEN 1:PAPER 2:LOCATE 13,24:PRINT"P L E A S E
  W A I T":PAPER 0
1850 FOR T= 1 TO 800
1860 NEXT T
1870 REM
1880 GOTO 1970
1890 REM
1900 FOR F=1 TO 3
1910 PEN 1:PAPER 2:LOCATE 4,25:PRINT"NO POSSIBLE M
OVE FOR THIS PLAYER"
1920 FOR G=1 TO 800
1930 NEXT G
1940 LOCATE 4,25:PAPER 2:PRINT"
                    "
1950 NEXT F
1960 GOTO 270
1970 FOR I=2 TO 9
1980 FOR J=2 TO 9
1990 LET W(I,J)=0
2000 NEXT J
2010 NEXT I
2020 LET AA=1
2030 LET BB=-1
2040 FOR I=2 TO 9
2050 FOR J=2 TO 9
2060 IF A(I,J)=0 THEN GOTO 2090
2070 LET W(I,J)=-8
2080 GOTO 2100
2090 GOSUB 2220
2100 NEXT J
```

```
2110 NEXT I
2120 FOR I=2 TO 9
2130 FOR J=2 TO 9
2140 IF W(I,J)=MX THEN GOTO 2180
2150 NEXT J
2160 NEXT I
2170 LOCATE 13,24:PAPER 2:PRINT"
        ":PAPER 0
2180 LOCATE 10,24:PAPER 2:PRINT"
        ":PAPER 0:
2190 PEN#2,0:LOCATE#2,7,6:PRINT#2,CHR$(I+63);RIGHT
$(STR$(J-1),1)
2200 LET A$=CHR$(I+63)+RIGHT$(STR$(J-1),1)
2210 GOTO 790
2220 FOR L=-1 TO 1
2230 FOR M=-1 TO 1
2240 IF A(I+L,J+M)=BB THEN GOTO 2290
2250 NEXT M
2260 NEXT L
2270 LET W(I,J)=-4
2280 RETURN
2290 REM
2300 FOR L=J+1 TO 10
2310 IF A(I,L)=0 THEN GOTO 2380
2320 IF A(I,L)=AA THEN GOTO 2340
2330 NEXT L
2340 IF L=J+1 THEN GOTO 2380
2350 FOR M=J+1 TO L-1
2360 LET W(I,J)=W(I,J)+1
2370 NEXT M
2380 FOR L=J-1 TO 1 STEP -1
2390 IF A(I,L)=0 THEN GOTO 2460
2400 IF A(I,L)=AA THEN GOTO 2420
2410 NEXT L
2420 IF L=J-1 THEN GOTO 2460
2430 FOR M=J-1 TO L+1 STEP-1
2440 LET W(I,J)=W(I,J)+1
2450 NEXT M
2460 FOR L=I+1 TO 10
2470 IF A(L,J)=0 THEN GOTO 2540
2480 IF A(L,J)=AA THEN GOTO 2500
2490 NEXT L
2500 IF L=I+1 THEN GOTO 2540
2510 FOR M=I+1 TO L-1
2520 LET W(I,J)=W(I,J)+1
2530 NEXT M
2540 FOR L=I-1 TO 1 STEP -1
2550 IF A(L,J)=0 THEN GOTO 2620
2560 IF A(L,J)=AA THEN GOTO 2580
2570 NEXT L
2580 IF L=I-1 THEN GOTO 2620
```

```
2590 FOR M=I-1 TO L+1 STEP -1
2600 LET W(I,J)=W(I,J)+1
2610 NEXT M
2620 LET Z=1
2630 LET M=J+Z
2640 LET L=I-Z
2650 IF L=1 OR M=10 THEN GOTO 2760
2660 IF A(L,M)=0 THEN GOTO 2760
2670 IF A(L,M)=AA THEN GOTO 2700
2680 LET Z=Z+1
2690 GOTO 2630
2700 IF Z=1 THEN GOTO 2760
2710 FOR V=1 TO Z-1
2720 LET L=I-V
2730 LET M=J+V
2740 LET W(I,J)=W(I,J)+1
2750 NEXT V
2760 LET Z=1
2770 LET L=I+Z
2780 LET M=J-Z
2790 IF L=10 OR M=1 THEN GOTO 2900
2800 IF A(L,M)=0 THEN GOTO 2900
2810 IF A(L,M)=AA THEN GOTO 2840
2820 LET Z=Z+1
2830 GOTO 2770
2840 IF Z=1 THEN GOTO 2900
2850 FOR V=1 TO Z-1
2860 LET L=I+V
2870 LET M=J-V
2880 LET W(I,J)=W(I,J)+1
2890 NEXT V
2900 LET Z=1
2910 LET L=I+Z
2920 LET M=J+Z
2930 IF L=10 OR M=10 THEN GOTO 3040
2940 IF A(L,M)=0 THEN GOTO 3040
2950 IF A(L,M)=AA THEN GOTO 2980
2960 LET Z=Z+1
2970 GOTO 2910
2980 IF Z=1 THEN GOTO 3040
2990 FOR V=1 TO Z-1
3000 LET L=I+V
3010 LET M=J+V
3020 LET W(I,J)=W(I,J)+1
3030 NEXT V
3040 LET Z=1
3050 LET L=I-Z
3060 LET M=J-Z
3070 IF L=1 OR M=1 THEN GOTO 3180
3080 IF A(L,M)=0 THEN GOTO 3180
3090 IF A(L,M)=AA THEN GOTO 3120
```

```
3100 LET Z=Z+1
3110 GOTO 3050
3120 IF Z=1 THEN GOTO 3180
3130 FOR V=1 TO Z-1
3140 LET L=I-V
3150 LET M=J-V
3160 LET W(I,J)=W(I,J)+1
3170 NEXT V
3180 IF W(I,J)=0 THEN LET W(I,J)=-8
3190 IF W(I,J)=-8 THEN RETURN
3200 IF I=J=2 THEN LET W(I,J)=W(I,J)+4
3210 IF I=J=9 THEN LET W(I,J)=W(I,J)+4
3220 IF I=2 AND J=9 THEN LET W(I,J)=W(I,J)+4
3230 IF I=9 AND J=2 THEN LET W(I,J)=W(I,J)+4
3240 IF I=2 THEN LET W(I,J)=W(I,J)+2
3250 IF I=9 THEN LET W(I,J)=W(I,J)+2
3260 IF J=2 THEN LET W(I,J)=W(I,J)+2
3270 IF J=9 THEN LET W(I,J)=W(I,J)+2
3280 IF I=3 OR I=8 THEN LET W(I,J)=W(I,J)-2
3290 IF J=3 OR J=8 THEN LET W(I,J)=W(I,J)-2
3300 IF W(I,J)>MX THEN LET MX=W(I,J)
3310 RETURN
3320 FOR X=29 TO 60
3330 NEXT X
3340 FOR Y=4 TO 37
3350 NEXT Y
3360 PAPER 2:CLS:BORDER 6:LOCATE 16,2:PEN 3:PRINT"
1 2 3 4 5 6 7 8"
3370 LET L=3
3380 FOR K=65 TO 72
3390 LET L=L+2
3400 LOCATE 13,L:PRINT CHR$(K)
3410 NEXT K
3420 FOR X=1 TO 18
3430 NEXT X
3440 FOR Y=12 TO 33
3450 NEXT Y
3460 PAPER 0:WINDOW#3,15,31,4,20:PAPER #3,0:CLS#3:
WINDOW#2,1,11,6,15:PAPER#2,3:PEN#2,0:CLS#2:LOCATE#
2,2,2:PRINT#2,"PLAYER="
3470 PLOT 0,160,0:DRAWR 174,0:DRAWR 0,159:DRAWR -1
74,0:DRAWR 0,-159
3480 LOCATE#2,2,6:PRINT#2,"MOVE:"
3490 LOCATE#2,2,8:PRINT#2,"LAST":LOCATE#2,3,9:PRIN
T#2,"MOVE:"
3500 RETURN
3510 REM
3520 FOR H=2 TO 9
3530 FOR G=2 TO 9
3540 PEN 1:PAPER 2:LOCATE 6,22:PRINT"W A I T   F O
 R   R E S U L T"
```

```
3550 IF A(H,G)=1 THEN LET JJ=JJ+1
3560 IF A(H,G)=-1 THEN LET QQ=QQ+1
3570 LOCATE 1,22:PAPER 2:PRINT"
                    "
3580 NEXT G
3590 NEXT H
3600 PAPER 0:BORDER 0:PEN 3:CLS
3610 IF JJ>QQ THEN PRINT N$;" HAS WON THE GAME,";J
J;" TO ";QQ
3620 IF QQ>JJ THEN PRINT M$;" HAS WON THE GAME,";Q
Q;" TO ";JJ
3630 IF QQ=JJ THEN PRINT"THE GAME WAS A DRAW"
3640 LOCATE 6,3:PRINT"DO YOU WANT ANOTHER GAME ?"
3650 IF INKEY$="" THEN GOTO 3650
3660 LET EN=0
3670 IF INKEY$="Y" THEN RUN
3680 IF INKEY$="N" THEN END
3690 GOTO 3660
3700 REM
3710 BORDER 0:PAPER 0:PLOT 0,0,3:CLS:PEN 3:FOR G=-
100 TO 176 STEP 4:MOVE G,250:TAG:PRINT" O T H E L
L O";
3720 NEXT G:TAGOFF
3730 FOR F=1 TO 20:NEXT F
3740 REM
3750 REM
3760 LET K=0
3770 REM
3780 INK 2,15:PEN 2:LOCATE 1,15:PRINT"NUMBER OF PL
AYERS ?
                    1(AGAINST THE COMPUTER) OR 2 ?"
3790 LET A$=INKEY$
3800 IF A$<"1" OR A$>"2" THEN GOTO 3790
3810 LET PL=VAL(A$)
3820 IF PL=2 THEN GOTO 3870
3830 PRINT:INPUT"NAME, PLEASE ";M$
3840 PRINT:INPUT"THIS PLAYER TO GO FIRST ";Q$:IF L
EFT$(Q$,1)="Y" THEN LET MOV=0 ELSE LET MOV=1
3850 LET N$="AMSTRAD"
3860 GOTO 3900
3870 PRINT:INPUT" PLAYER 1 ";M$
3880 PRINT:INPUT"THIS PLAYER TO GO FIRST ";Q$:IF L
EFT$(Q$,1)="Y" THEN LET MOV=0 ELSE LET MOV=1
3890 PRINT:INPUT" PLAYER 2 ";N$
3900 CLS
3910 INK 2,6
3920 GOSUB 3320
3930 RETURN
```

8. Noughts and Crosses

One of the oldest games in the world, which has passed the time for millions of people in the silent surroundings of dentists' waiting rooms, must surely be Noughts and Crosses.

The game is presented here in a colourful and graphical manner, with the computer trained to handle the crafty moves which you may make when trying to win. So play carefully and choose your moves wisely.

Playing instructions

When the game is executed you are offered the first move. If you accept you may choose a level of play by entering a number in the range 1-5. Level 1 is the easiest and is intended for novices, but if you think you can outwit the computer then try level 5. If you have declined the initial offer then this stage is omitted because the computer will automatically play to win. The computer will show you your symbol (nought) and its own (cross) and the game commences.

1	2	3
4	5	6
7	8	9

The game progresses as you take alternate turns at positioning your symbol on a three by three grid in an attempt to be the first player to obtain a straight line of symbols. It is possible to finish with all the squares occupied and no straight lines of either symbol, and in such a situation the match is drawn. To position your symbol you must enter the number corresponding to the square you wish to occupy on the grid. These are shown in the diagram on p.132.

The program

The heart of this program is the routine which determines if there is a chance for the computer to complete a winning line or a need to block that of its opponent, and this can be found in lines 1690-2120. This section must be entered with extra care to avoid an error in the code which would result in ridiculous moves.

The flowchart

The flowchart on p.134 represents the general operation of the program.

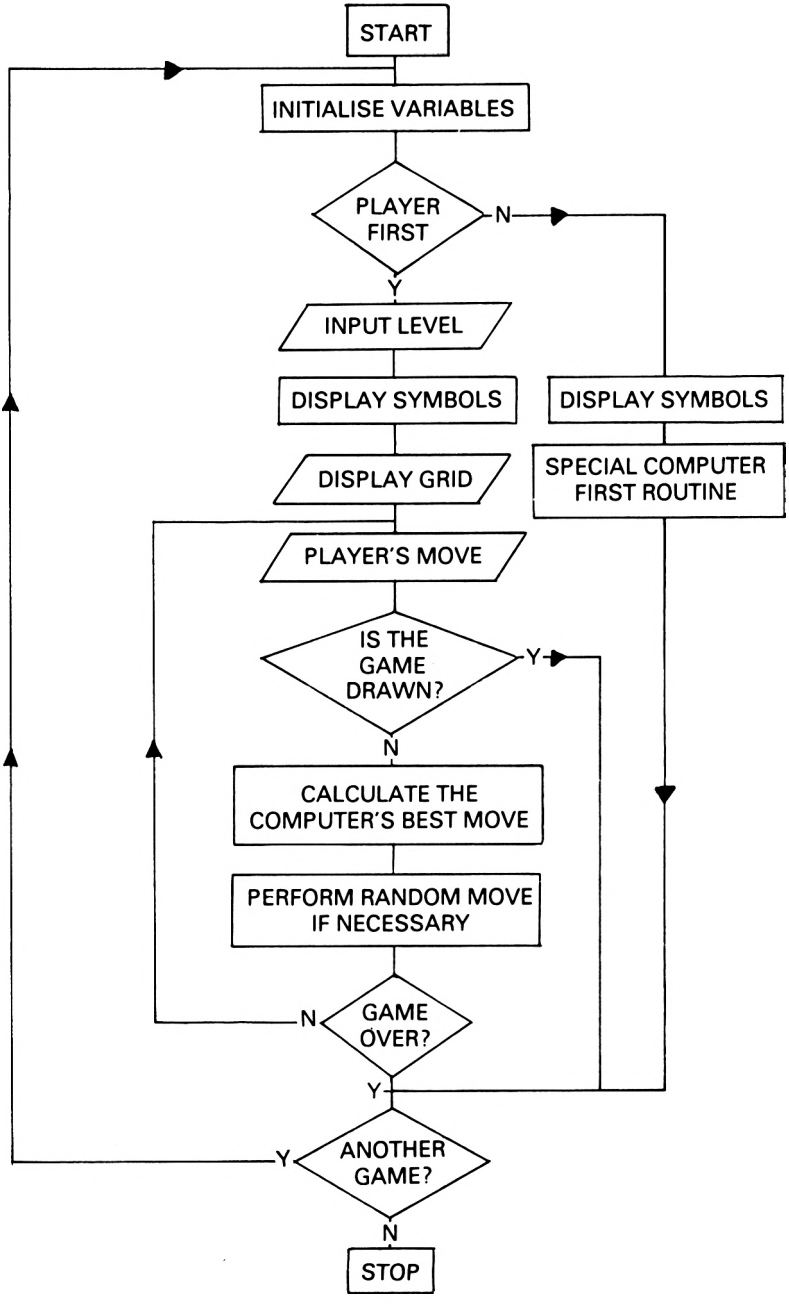
Variables

The following table represents the major variables used in the program and should help you to understand how the program operates.

B()	Array containing the current board position
F$	Player with first move
S	Current square to be occupied
T	Game over
DR	Game drawn
J	Victorious player
K	Player whose pieces are being tested
PA$	Another game
L$	Level of play in the range 1-5
L	Associated variable to L$ in the range 0.6-1

Comments on the program

The program falls into two main sections depending on who has


START
INITIALISE VARIABLES
PLAYER FIRST
N
Y
INPUT LEVEL
DISPLAY SYMBOLS
DISPLAY GRID
PLAYER'S MOVE
IS THE GAME DRAWN?
Y
N
CALCULATE THE COMPUTER'S BEST MOVE
PERFORM RANDOM MOVE IF NECESSARY
GAME OVER?
N
Y
ANOTHER GAME?
Y
N
STOP
DISPLAY SYMBOLS
SPECIAL COMPUTER FIRST ROUTINE

the first move, with various subroutines being called from them both. They are different in design because if the computer takes the first move it will attack immediately, but if the player goes first the computer will consider defence as the natural course of action. This is not to say that it will not play to win in the second case, since it will attack at the first convenient moment.

The subroutines are clearly labelled in the listing and a breakdown of them is shown below.

Lines 10-100
In this routine the variables are initialised with the contents of B() being set to 0. During the game, if the player occupies a given square the variable will hold the value 1; if the computer occupies it then the value will be 2. The colours are defined, the windows created and the subroutine to construct the user-defined graphics is called.

Line 110
Calls the subroutine which displays the instructions.

Lines 120-190
By calling several routines, the screen is designed ready for the game to commence.

Line 200
If the computer is to go first the routine starting at line 490 is selected.

Lines 210-480
This routine deals with the game when the player has the first move. After the first two squares have been selected, the program uses a loop to avoid re-writing the same code three times. Before the computer's move is finally accepted, the program branches to the subroutine at line 2410 which is used to decide if a random move is to be made. This is one method of including different levels of play and is explained in the appropriate section below.

Lines 490-1180
This is the section of the program which handles the attack when the computer goes first. Throughout this routine the computer demonstrates artificial intelligence by logically selecting the best possible move.

Lines 1190-1300
This is where the grid is produced on the screen.

Lines 1310-1400
This routine will draw a cross at any position on the screen. To obtain the correct place, the (x,y) co-ordinates of the top left-hand corner are given in the variables X and Y.

Lines 1410-1500
This works in the same way as the above routine, but will draw the nought.

Lines 1510-1540
This is used to calculate the values of X and Y for the above two routines.

Lines 1550-1680
A subroutine to test the possible winning combinations to decide if the game has been won.

Lines 1690-2120
This routine examines the current board position and decides on the best move for the computer. This is another routine which demonstrates a degree of intelligence.

Lines 2130-2260
When the game has finished, this routine will display the result and ask whether you wish to play a further game.

Lines 2270-2400
This is the subroutine which asks the player for his move and decides whether or not his response is legal. It uses different colours to display appropriate messages in one of the windows on the screen.

Lines 2410-2430
This small section allows for five levels of play to be available if the player opts for the first move. It is a simple routine which produces a random number and performs a test to see if a random move should be selected.

When control passes to this section the variable S already contains the best move for the computer and L is a number between 0.6 and 1 representing the level of play. If a random number (0-0.9999) is smaller than L, then no action is taken, but if it is larger then a legal random move is made.

Clearly if L=1 then all random numbers in the range 0-.0.9999

are smaller and so the computer will always make the best move. This corresponds to level 5.

Lines 2440-2500
Removes the symbol from the supply when they are transferred to the grid.

Lines 2510-2560
Draws the borders around the windows.

Lines 2570-2710
This creates the user-defined graphics used throughout the program.

Lines 2720-2790
This routine produces a heading without using a window. It is not included to show you how much extra work is needed when a window is not used (although it does), but is employed because it is the only way to get the heading central.

Lines 2800-2980
The instructions routine which offers the first move and the level of play.

The listing

The listing should be entered very carefully and then saved onto cassette as outlined in the Introduction.

The game should be entered and played in the caps lock mode. All spaces within the text should be entered as they appear.

```
10 MODE 0
20 DIM B(9)
30 FOR I=1 TO 9:LET B(I)=0:NEXT I
40 LET A=0:LET DR=0
50 INK 0,20:INK 1,6:INK 2,2:INK 3,0:INK 4,24:INK 5
,12:INK 6,4:INK 7,20,6:INK 8,20,2
60 WINDOW #1,7,14,6,17:PAPER #1,0
70 WINDOW #2,2,5,6,24:PAPER #2,0
80 WINDOW #3,16,19,6,24:PAPER #3,0
90 WINDOW #4,7,14,20,23:PAPER #4,0:PEN #4,7
100 GOSUB 2570
110 GOSUB 2800
120 BORDER 4:PAPER 4:CLS:CLS #1:CLS #2:CLS #3:CLS
#4
```

```
130 GOSUB 2510
140 GOSUB 2720
150 GOSUB 1190
160 FOR Y=301 TO 93 STEP -52
170 LET X=64:GOSUB 1410
180 LET X=514:GOSUB 1310
190 NEXT Y
200 IF F$="N" THEN GOTO 490
210 GOSUB 2270
220 X=64:Y=93:GOSUB 2440
230 IF S<>5 THEN LET S=5:GOTO 250
240 LET S=1+2*INT(5*RND(1)):IF S=5 THEN GOTO 240
250 FOR JJ=1 TO 500:NEXT JJ
260 GOSUB 2410:GOSUB 1510:GOSUB 1310:LET X=514:LET
 Y=93:GOSUB 2440:LET B(S)=2
270 FOR A=1 TO 3
280 GOSUB 2270
290 LET X=64:LET Y=93+52*A:GOSUB 2440
300 GOSUB 1550:IF T=1 THEN GOTO 2130
310 LET K=2:GOSUB 1690
320 IF S<>0 THEN GOSUB 2410:LET B(S)=2:GOSUB 1510:
GOSUB 1310:GOTO 400
330 LET K=1:GOSUB 1690
340 IF S<>0 THEN GOSUB 2410:LET B(S)=2:GOSUB 1510:
GOSUB 1310:GOTO 400
350 LET S=1+INT(9*RND(1))
360 IF B(S)=0 THEN GOTO 390
370 GOTO 350
380 GOSUB 2410
390 LET B(S)=2:GOSUB 1510:GOSUB 1310
400 LET X=514:LET Y=93+52*A:GOSUB 2440
410 GOSUB 1550
420 IF T=1 THEN GOTO 2130
430 NEXT A
440 GOSUB 2270:GOSUB 1510:GOSUB 1410
450 LET X=64:LET Y=301:GOSUB 2440
460 GOSUB 1550
470 IF T=0 THEN LET DR=1
480 GOTO 2130
490 REM ** COMPUTER FIRST **
500 LET S=1+INT(9*RND(1)):IF S/2=INT(S/2) THEN GOT
O 500
510 LET B(S)=2
520 GOSUB 1510:GOSUB 1310:LET X=514:LET Y=93:GOSUB
 2440
530 IF S<>5 THEN GOTO 1030
540 GOSUB 2270
550 LET X=64:LET Y=93:GOSUB 2440
560 IF S/2<>INT(S/2) THEN GOTO 710
570 IF S=2 OR S=4 THEN LET S=9
580 IF S=6 OR S=8 THEN LET S=1
```

```
590 LET B(S)=2
600 GOSUB 1510:GOSUB 1310:LET X=514:LET Y=145:GOSU
B 2440
610 GOSUB 2270
620 LET X=64:LET Y=145:GOSUB 2440
630 LET K=2:GOSUB 1690:IF S<>0 THEN GOTO 650
640 LET K=1:GOSUB 1690:IF S<>0 THEN GOTO 650
650 GOSUB 1510:GOSUB 1310:LET B(S)=2:LET X=514:LET
 Y=197:GOSUB 2440
660 GOSUB 1550:IF T=1 THEN GOTO 2130
670 GOSUB 2270:LET X=64:LET Y=197:GOSUB 2440
680 LET K=2:GOSUB 1690:LET B(S)=2:GOSUB 1510:GOSUB
 1310
690 LET X=514:LET Y=249:GOSUB 2440
700 GOSUB 2130
710 LET S=10-S:LET S1=S:LET B(S)=2:GOSUB 1510:GOSU
B 1310
720 LET X=514:LET Y=145:GOSUB 2440
730 GOSUB 2270
740 LET X=64:LET Y=145:GOSUB 2440
750 LET K=1:GOSUB 1690:IF S<>0 THEN GOTO 930
760 IF S1<>9 THEN GOTO 790
770 IF B(6)=1 THEN LET S=8
780 IF B(8)=1 THEN LET S=6
790 IF S1<>7 THEN GOTO 820
800 IF B(4)=1 THEN LET S=8
810 IF B(8)=1 THEN LET S=4
820 IF S1<>3 THEN GOTO 850
830 IF B(2)=1 THEN LET S=6
840 IF B(6)=1 THEN LET S=2
850 IF S1<>1 THEN GOTO 880
860 IF B(2)=1 THEN LET S=4
870 IF B(4)=1 THEN LET S=2
880 LET B(S)=2:GOSUB 1510:GOSUB 1310:LET X=514:LET
 Y=197:GOSUB 2440
890 GOSUB 2270:LET X=64:LET Y=197:GOSUB 2440
900 LET K=2:GOSUB 1690:LET B(S)=2:GOSUB 1510:GOSUB
 1310
910 LET X=514:LET Y=249:GOSUB 2440
920 GOSUB 1550:GOSUB 2130
930 LET K=1:GOSUB 1690:LET B(S)=2:GOSUB 1510:GOSUB
 1310:LET X=514:LET Y=197:GOSUB 2440
940 GOSUB 2270:LET X=64:LET Y=197:GOSUB 2440
950 LET K=2:GOSUB 1690:IF S<>0 THEN GOTO 980
960 LET K=1:GOSUB 1690:IF S<>0 THEN GOTO 980
970 LET S=1+INT(9*RND(1)):IF B(S)<>0 THEN GOTO 970
980 LET B(S)=2:GOSUB 1510:GOSUB 1310:LET X=514:LET
 Y=249:GOSUB 2440:GOSUB 1550:IF T=1 THEN GOTO 2130
990 GOSUB 2270:LET X=64:LET Y=249:GOSUB 2440
1000 FOR A=1 TO 9:IF B(A)=0 THEN LET S=A
1010 NEXT A
```

```
1020 LET B(S)=2:GOSUB 1510:GOSUB 1310:LET X=514:LE
T Y=301:GOSUB 2440:LET DR=1:GOTO 2130
1030 LET S1=S:GOSUB 2270:LET X=64:LET Y=93:GOSUB 2
440
1040 IF S<>5 THEN LET S=5:GOTO 1060
1050 LET S=10-S1
1060 LET B(S)=2:GOSUB 1510:GOSUB 1310:LET X=514:LE
T Y=145:GOSUB 2440
1070 FOR B=1 TO 3
1080 GOSUB 2270:LET X=64:LET Y=93+52*B:GOSUB 2440
1090 LET K=2:GOSUB 1690:IF S<>0 THEN GOTO 1130
1100 LET K=1:GOSUB 1690:IF S<>0 THEN GOTO 1130
1110 IF B=1 AND B(5)=2 THEN GOTO 760
1120 LET S=1+INT(9*RND(1)):IF B(S)<>0 THEN GOTO 11
20
1130 LET B(S)=2:GOSUB 1510:GOSUB 1310:LET X=514:LE
T Y=145+52*B:GOSUB 2440
1140 GOSUB 1550:IF T=1 THEN GOTO 2130
1150 NEXT B
1160 LET DR=1:GOTO 2130
1170 IF INKEY$="" THEN GOTO 1170
1180 END
1190 REM ** DRAW BOARD **
1200 PLOT 0,0,5:FOR I=1 TO 9
1210 IF I<4 THEN MOVE 204+68*(I-1),285:TAG:PRINT I
;:GOTO 1240
1220 IF I>6 THEN MOVE 204+68*(I-7),181:TAG:PRINT I
;:GOTO 1240
1230 MOVE 204+68*(I-4),233:TAG:PRINT I;
1240 NEXT I
1250 PLOT 286,300,3:DRAW 286,150:DRAW 287,150:DRAW
 287,300
1260 PLOT 354,300,3:DRAW 354,150:DRAW 355,150:DRAW
 355,300
1270 PLOT 220,252,3:DRAW 420,252:DRAW 420,250:DRAW
 220,250
1280 PLOT 220,200,3:DRAW 420,200:DRAW 420,199:DRAW
 220,199
1290 PLOT 0,0,4
1300 RETURN
1310 REM ** DRAW CROSS **
1320 PLOT 0,0,1
1330 MOVE X,Y:TAG
1340 PRINT CHR$(200);CHR$(201);
1350 MOVE X,Y-16
1360 PRINT CHR$(202);CHR$(203);
1370 MOVE X,Y-32
1380 PRINT CHR$(204);CHR$(205);
1390 TAGOFF
1400 RETURN
1410 REM ** DRAW NOUGHT **
```

```
1420 PLOT 0,0,2
1430 MOVE X,Y:TAG
1440 PRINT CHR$(206);CHR$(207);
1450 MOVE X,Y-16
1460 PRINT CHR$(208);CHR$(209);
1470 MOVE X,Y-32
1480 PRINT CHR$(210);CHR$(211);
1490 TAGOFF
1500 RETURN
1510 REM ** POSITION OF SYMBOL **
1520 LET X=152+68*(S-3*INT((S-1)/3))
1530 LET Y=301-52*INT((S-1)/3)
1540 RETURN
1550 REM ** RESULT ? **
1560 LET DR=0:LET T=0
1570 FOR J=1 TO 2
1580 IF B(1)=J AND B(2)=J AND B(3)=J THEN GOTO 168
0
1590 IF B(1)=J AND B(5)=J AND B(9)=J THEN GOTO 168
0
1600 IF B(1)=J AND B(4)=J AND B(7)=J THEN GOTO 168
0
1610 IF B(2)=J AND B(5)=J AND B(8)=J THEN GOTO 168
0
1620 IF B(3)=J AND B(6)=J AND B(9)=J THEN GOTO 168
0
1630 IF B(3)=J AND B(5)=J AND B(7)=J THEN GOTO 168
0
1640 IF B(4)=J AND B(5)=J AND B(6)=J THEN GOTO 168
0
1650 IF B(7)=J AND B(8)=J AND B(9)=J THEN GOTO 168
0
1660 NEXT J
1670 RETURN
1680 LET T=1:RETURN
1690 REM ** SELECT MOVE **
1700 LET S=0:IF B(1)=1 AND B(9)=1 AND B(5)=2 AND A
=1 THEN LET S=2:GOTO 2120
1710 IF B(3)=1 AND B(7)=1 AND B(5)=2 AND A=1 THEN
LET S=8:GOTO 2120
1720 IF B(1)=1 AND B(6)=1 AND B(5)=2 AND A=1 THEN
LET S=9:GOTO 2120
1730 IF B(1)=1 AND B(8)=1 AND B(5)=2 AND A=1 THEN
LET S=9:GOTO 2120
1740 IF B(1)=K AND B(2)=K AND B(3)=0 THEN LET S=3:
GOTO 2120
1750 IF B(2)=1 AND B(4)=1 AND B(5)=2 AND A=1 THEN
LET S=1:GOTO 2120
1760 IF B(2)=1 AND B(6)=1 AND B(5)=2 AND A=1 THEN
LET S=3:GOTO 2120
1770 IF B(8)=1 AND B(6)=1 AND B(5)=2 AND A=1 THEN
```

```
LET S=9:GOTO 2120
1780 IF B(8)=1 AND B(4)=1 AND B(5)=2 AND A=1 THEN
LET S=7:GOTO 2120
1790 IF B(1)=K AND B(3)=K AND B(2)=0 THEN LET S=2:
GOTO 2120
1800 IF B(1)=K AND B(4)=K AND B(7)=0 THEN LET S=7:
GOTO 2120
1810 IF A=1 AND B(5)=1 AND B(9)=1 AND B(1)=2 AND B
(7)=0 THEN LET S=7:GOTO 2120
1820 IF A=1 AND B(5)=1 AND B(7)=1 AND B(3)=2 AND B
(9)=0 THEN LET S=9:GOTO 2120
1830 IF A=1 AND B(5)=1 AND B(3)=1 AND B(7)=2 AND B
(1)=0 THEN LET S=1:GOTO 2120
1840 IF A=1 AND B(5)=1 AND B(1)=1 AND B(9)=2 AND B
(3)=0 THEN LET S=3:GOTO 2120
1850 IF B(1)=K AND B(5)=K AND B(9)=0 THEN LET S=9:
GOTO 2120
1860 IF B(1)=K AND B(7)=K AND B(4)=0 THEN LET S=4:
GOTO 2120
1870 IF B(1)=K AND B(9)=K AND B(5)=0 THEN LET S=5:
GOTO 2120
1880 IF B(3)=1 AND B(4)=1 AND B(5)=2 AND A=1 THEN
LET S=7:GOTO 2120
1890 IF B(3)=1 AND B(8)=1 AND B(5)=2 AND A=1 THEN
LET S=7:GOTO 2120
1900 IF B(2)=K AND B(3)=K AND B(1)=0 THEN LET S=1:
GOTO 2120
1910 IF B(2)=K AND B(5)=K AND B(8)=0 THEN LET S=8:
GOTO 2120
1920 IF B(7)=1 AND B(2)=1 AND B(5)=2 AND A=1 THEN
LET S=3:GOTO 2120
1930 IF B(7)=1 AND B(6)=1 AND B(5)=2 AND A=1 THEN
LET S=3:GOTO 2120
1940 IF B(2)=K AND B(8)=K AND B(5)=0 THEN LET S=5:
GOTO 2120
1950 IF B(3)=K AND B(5)=K AND B(7)=0 THEN LET S=7:
GOTO 2120
1960 IF B(9)=1 AND B(2)=1 AND B(5)=2 AND A=1 THEN
LET S=1:GOTO 2120
1970 IF B(9)=1 AND B(4)=1 AND B(5)=2 AND A=1 THEN
LET S=1:GOTO 2120
1980 IF B(3)=K AND B(6)=K AND B(9)=0 THEN LET S=9:
GOTO 2120
1990 IF B(3)=K AND B(7)=K AND B(5)=0 THEN LET S=5:
GOTO 2120
2000 IF B(3)=K AND B(9)=K AND B(6)=0 THEN LET S=6:
GOTO 2120
2010 IF B(4)=K AND B(5)=K AND B(6)=0 THEN LET S=6:
GOTO 2120
2020 IF B(4)=K AND B(6)=K AND B(5)=0 THEN LET S=5:
GOTO 2120
```

```
2030 IF B(4)=K AND B(7)=K AND B(1)=0 THEN LET S=1:
GOTO 2120
2040 IF B(6)=K AND B(5)=K AND B(4)=0 THEN LET S=4:
GOTO 2120
2050 IF B(6)=K AND B(9)=K AND B(3)=0 THEN LET S=3:
GOTO 2120
2060 IF B(7)=K AND B(5)=K AND B(3)=0 THEN LET S=3:
GOTO 2120
2070 IF B(7)=K AND B(8)=K AND B(9)=0 THEN LET S=9:
GOTO 2120
2080 IF B(7)=K AND B(9)=K AND B(8)=0 THEN LET S=8:
GOTO 2120
2090 IF B(8)=K AND B(9)=K AND B(7)=0 THEN LET S=7:
GOTO 2120
2100 IF B(8)=K AND B(5)=K AND B(2)=0 THEN LET S=2:
GOTO 2120
2110 IF B(9)=K AND B(5)=K AND B(1)=0 THEN LET S=1:
GOTO 2120
2120 RETURN
2130 REM ** GAME OVER **
2140 PEN #4,3
2150 CLS #4:PRINT #4
2160 IF DR=1 THEN PRINT #4,"  MATCH   DRAWN":GOTO
2190
2170 IF J=1 THEN PRINT #4," PLAYER   WINS":GOTO 21
90
2180 PRINT #4,"COMPUTER  WINS"
2190 FOR JJ=1 TO 3000:NEXT JJ
2200 CLS #4:PRINT #4
2210 PRINT #4,"  PLAY   AGAIN?"
2220 LET PA$=INKEY$:IF PA$="" THEN GOTO 2220
2230 IF PA$="N" THEN CLS:END
2240 FOR I=1 TO 9:LET B(I)=0:NEXT I
2250 GOSUB 2800
2260 GOTO 120
2270 REM ** MESSAGE **
2280 CLS #4
2290 PRINT #4
2300 PEN #4,8:PRINT #4,"PLAYER'S  TURN"
2310 LET S$=INKEY$:IF S$="" THEN GOTO 2310
2320 IF ASC(S$)<49 OR ASC(S$)>57 THEN GOTO 2310
2330 IF B(VAL(S$))<>0 THEN GOTO 2310
2340 LET S=VAL(S$)
2350 LET B(S)=1:GOSUB 1510:GOSUB 1410
2360 CLS #4
2370 PRINT #4
2380 PEN #4,7:PRINT #4,"COMPUTERTO  MOVE"
2390 FOR JJ=1 TO 1000:NEXT JJ
2400 RETURN
2410 IF RND(1)<L THEN RETURN
2420 LET S=1+INT(9*RND(1)):IF B(S)<>0 THEN GOTO 24
```

```
20
2430 RETURN
2440 REM ** ERASE **
2450 TAG
2460 MOVE X,Y:PRINT "  ";
2470 MOVE X,Y-16:PRINT "  ";
2480 MOVE X,Y-32:PRINT "  ";
2490 TAGOFF
2500 RETURN
2510 REM ** BORDERS **
2520 PLOT 30,320,3:DRAW 30,14:DRAW 162,14:DRAW 162
,320:DRAW 30,320
2530 PLOT 478,320,3:DRAW 478,14:DRAW 610,14:DRAW 6
10,320:DRAW 478,320
2540 PLOT 190,320,3:DRAW 450,320:DRAW 450,126:DRAW
 190,126:DRAW 190,320
2550 PLOT 190,96,3:DRAW 190,30:DRAW 450,30:DRAW 45
0,96:DRAW 190,96
2560 RETURN
2570 REM ** DEFINE CHARS **
2580 SYMBOL AFTER 200
2590 SYMBOL 200,&0,&0,&0,&0,&20,&30,&38,&3C
2600 SYMBOL 201,&0,&0,&0,&0,&4,&C,&1C,&3C
2610 SYMBOL 202,&1E,&F,&7,&3,&3,&7,&F,&1E
2620 SYMBOL 203,&78,&F0,&E0,&C0,&C0,&E0,&F0,&78
2630 SYMBOL 204,&3C,&38,&30,&20,&0,&0,&0,&0
2640 SYMBOL 205,&3C,&1C,&C,&4,&0,&0,&0,&0
2650 SYMBOL 206,&0,&0,&0,&0,&3F,&3F,&3F,&38
2660 SYMBOL 207,&0,&0,&0,&0,&FC,&FC,&FC,&1C
2670 SYMBOL 208,&38,&38,&38,&38,&38,&38,&38,&38
2680 SYMBOL 209,&1C,&1C,&1C,&1C,&1C,&1C,&1C,&1C
2690 SYMBOL 210,&38,&3F,&3F,&3F,&0,&0,&0,&0
2700 SYMBOL 211,&1C,&FC,&FC,&FC,&0,&0,&0,&0
2710 RETURN
2720 REM ** HEADING **
2730 MOVE 16,380:TAG:PRINT"                    ";
2740 MOVE 16,360:TAG:PRINT"                    ";
2750 PLOT 0,0,6
2760 MOVE 16,370:TAG:PRINT" NOUGHTS & CROSSES ";
2770 TAGOFF
2780 PLOT 16,382,3:DRAW 16,344:DRAW 626,344:DRAW 6
26,382:DRAW 16,382
2790 RETURN
2800 REM ** INSTRUCTIONS **
2810 BORDER 4:PAPER 4:PEN 1:CLS
2820 FOR J=1 TO 5:PRINT:NEXT J
2830 GOSUB 2720
2840 PRINT "  DO YOU  WANT THE":PRINT "  FIRST MOV
E (Y/N) "
2850 LET F$=INKEY$:IF F$="" THEN GOTO 2850
2860 IF F$<>"Y" AND F$<>"N" THEN GOTO 2850
```

```
2870 IF F$="N" THEN PRINT:PRINT:PRINT:GOTO 2930
2880 PRINT:PRINT" ENTER LEVEL 1-5  ";
2890 LET L$=INKEY$:IF L$<"1" OR L$>"5" THEN GOTO 2
890
2900 PEN 3:PRINT L$:PEN 1
2910 LET L=0.5+VAL(L$)/10
2920 PRINT
2930 PRINT:PRINT"  YOU ARE  PLAYING":PRINT:PRINT:P
RINT:PRINT
2940 LET X=288:LET Y=194:GOSUB 1410
2950 PRINT:PRINT"  THE  COMPUTER IS"
2960 LET Y=100:GOSUB 1310
2970 FOR J=1 TO 3000:NEXT J
2980 RETURN
```

9. Cribbage

If you are a card-playing fanatic, this is your opportunity to test your ability against the computer in one of the most skilful games for two players.

As the game progresses it should become obvious that the computer is no easy adversary. The program is designed to detect many of the crafty tricks that you may play for extra points, and the computer also knows a few tricks of its own. So good luck, play fair and with lots of practice you might just win a game if you are lucky!

Playing instructions

When the game is executed the cards will be cut to see who has the first crib and the computer will deal out the cards.

From this point on you refer to your cards by typing a number as indicated on the screen, and if at any time during the 'play for points', you wish to pass, then pressing the P-key will cause the computer to take its next turn. The game follows the normal sequence of Cribbage and it is assumed that the normal rules of the game are fully understood.

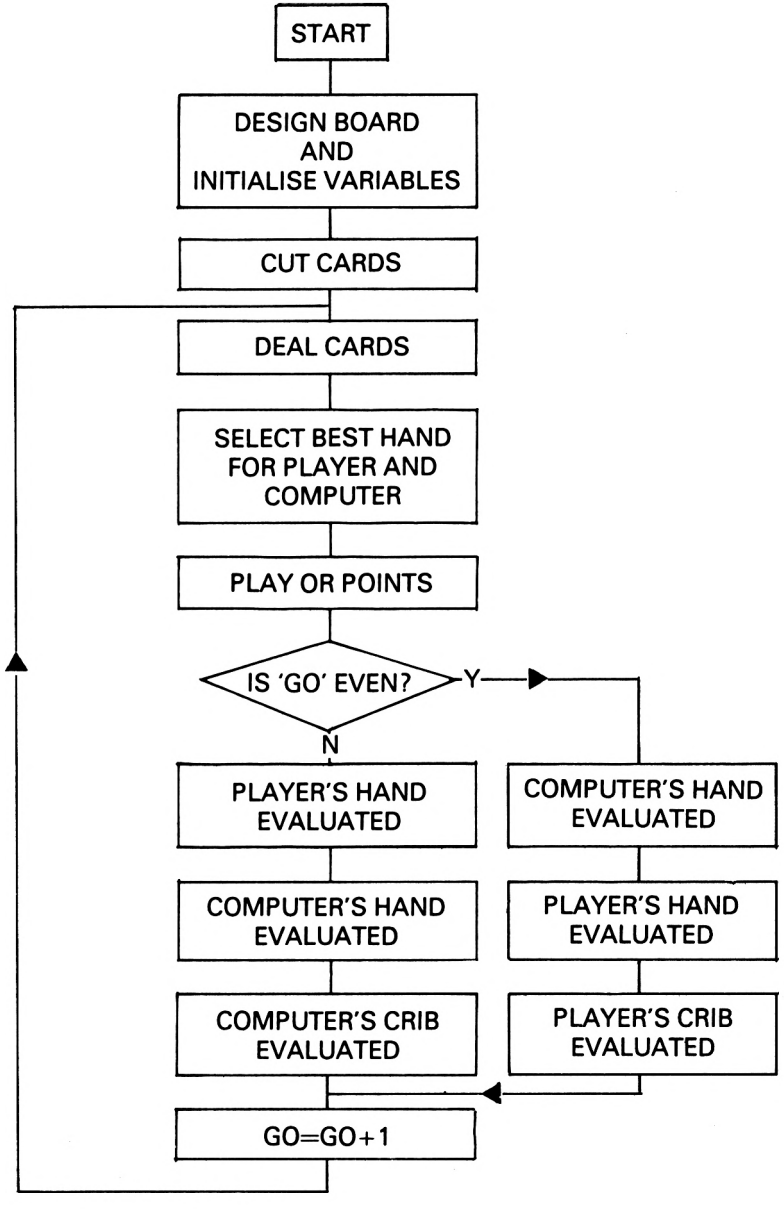
The program

Unlike many of the games in this book, Cribbage does not use the system of weightings, but considers every available possibility and selects that which gives the maximum number of points. The computer therefore plays a very safe game and will always choose its cards cautiously rather than gambling on a particular cut of the cards.

Over an extensive testing period this method has proved to be most satisfactory and although it is possible to win by a rather large margin on occasions, over a series of games the computer will usually have the advantage.

Flowchart

The flowchart represents the general operation of the program:



Variables

The following table represents the major variables used in the program and should be of assistance in trying to understand how the program operates.

A$()	Array storing cards for current round
B$()	Copy of array A$() for sorting
C$()	Cribbage hand
F$()	Cut card
I$()	Computer's hand
P$()	Player's hand
SCOREC	Computer's score
SCOREP	Player's score
GO	Counter for number of hands played
PO	Counter for play situation
CN	Number of cards for display
Q$()	Suit of cards
I()	Colour of cards (red or black)
S	Score of hand calculated by evaluator

Comments

The program is highly structured, with the majority of the code being contained within subroutines. Each subroutine is clearly labelled in the listing and a breakdown of each routine is shown below.

Lines 10-80
The variables are initialised and the screen colours are defined as explained in Chapter 2. Note that the global colour (INK 0) is set to 26 so that any characters printed using the TAG command are produced on a white background.

Lines 90-1260
This represents the whole of the main program which keeps control of the game, making use of the subroutines which constitute the largest part of the program.

Lines 1270-1440
This routine is used to update the scoreboard by moving the pegs by an appropriate amount. A check is also made in this routine to see if a player has 121 or more points, in which case the game is won and the end routine is called.

Lines 1450-1650
This routine is used during the 'play for points' section of the game. The player and the computer are each given alternate turns and the appropriate evaluation and score routines are summoned.

Lines 1660-2000
A routine to control the player's turn when playing for points, including legality checks and any necessary updating of the score.

Lines 2010-2340
Routine to control the computer's turn when playing for points.

Lines 2350-2590
This represents one of the two sections which give the program a measure of intelligence. It is used to find the player's score and also to evaluate the best possible card for the computer to play.

Lines 2600-2800
A complex routine to draw a single card on the screen. The value of the card is first checked and the appropriate graphical representation is produced. If the card is a Jack, Queen or King then an enlarged letter is produced by using a number of CHR$ variables.

Lines 2810-2870
A short routine which numbers the cards in order to simplify the player's selection process.

Lines 2880-2910
A routine to clear the window used for reporting messages throughout the game.

Lines 2920-2950
Similar to the above routine, but used to clear the window in which the cards are displayed.

Lines 2960-3060
Draws the outline so that a hand of cards can be displayed in the bottom window. By changing the value of CN, any number of cards can be constructed.

Lines 3070-3260
A routine which considers a hand of cards and assigns values to Q$() and I() concerning the suits and colours, and then constructs the cards in the appropriate window.

Lines 3270-3290
Produces an outline for a single card.

Lines 3300-3350
When the pack of cards is displayed on the screen this routine is used to produce the shading on the top card. By using the appropriate CHR$ we obtain a stippled pattern of red and white creating an illusion of a different colour.

Lines 3360-3990
This is the second area of the program which displays a measure of artificial intelligence. The routine is used to evaluate the total in each of the three hands, and is also used to find the best possible combination of cards for the computer. It works by considering all of the different methods of scoring points (pairs, fifteens, runs, etc.) and then obtains a total score, 'S', for that combination of cards.

Lines 4000-4220
This is the control routine for the selection of the computer's hand and is used in conjunction with the evaluator described above.

Lines 4230-4370
The game-over routine, which displays the final positions and reports the winner.

Lines 4380-4660
A very useful routine which you may find helpful in the design of other card games. It is used to deal or select twelve cards from a standard pack of fifty-two. When a card has been selected it is assigned to the array A$() in the form:

H04	four of hearts
S13	king of spades

A quick check is then made of the cards previously selected to ensure that no card has been dealt twice, and the loop continues.

Lines 4670-4710
A simple routine to transfer the contents of the array A$() into B$() so that the other routines can operate without corrupting A$().

Lines 4720-4780
The original routine used to display the cribbage board on the screen.

Lines 4790-4910
This routine is called very early in the program and is used to define the windows and the user-defined graphics.

The listing

The listing should be entered very carefully and then saved on to cassette as outlined in the Introduction.

The game should be entered and played in the caps lock mode. All spaces within the text should be entered as they appear.

```
10 MODE 1:BORDER 18:INK 0,26:INK 1,0:INK 2,6:INK
3,18:PAPER 3:CLS:PAPER 0
20 LET RC=0:LET RP=0:LET GO=INT(RND(1)*2)+1:LET FL
=0
30 GOSUB 4790
40 LET SC1=18:LET SC2=375:LET SP1=18:LET SP2=295
50 LET SCOREP=0:LET SCOREC=0
60 RANDOMIZE TIME
70 DIM C(5),G$(8),D$(4),P(8),I$(5),F$(1),A$(12),H$
(5),B$(12),P$(5),Q$(8),I(10),R$(8),C$(5)
80 LET F$(1)="999"
90 GOSUB 1270
100 LOCATE#2,2,2:PRINT#2, "DEALING"
110 LOCATE#2,3,3:PRINT#2,"CARDS"
120 GOSUB 4380
130 GOSUB 2880:IF GO/2<>INT(GO/2) THEN LOCATE#2,2,
2:PRINT#2,"PLAYERS":LOCATE#2,4,3:PRINT#2,"CRIB"
140 IF GO/2=INT(GO/2) THEN MOVE 488,390:TAG:PRINT
"COMPUTERS";:TAGOFF:LOCATE#2,4,3:PRINT#2,"CRIB"
150 LET S=1:LET F=6:GOSUB 4520
160 LET S=7:LET F=12:GOSUB 4520:GOSUB 4670
170 FOR I=1 TO 6:LET A$(I)=B$(I+6):NEXT I:LET CN=6
:LET C1=1:LET C2=6:GOSUB 2960
180 GOSUB 3070
190 GOSUB 3200
200 GOSUB 2600
210 LET X1=486:LET Y1=230:GOSUB 3300
220 GOSUB 3270
230 GOSUB 2880:LOCATE #2,2,2:PRINT #2,"COMPUTER":L
OCATE #2,2,3:PRINT #2,"THINKING":LOCATE #2,3,5:PRI
NT #2,"PLEASE":LOCATE #2,4,6:PRINT#2,"WAIT":FOR I=
1 TO 6:LET A$(I)=B$(I):NEXT I:GOSUB 4000
```

```
240 GOSUB 2880
250 LOCATE #2,3,2:PRINT #2,"ENTER":LOCATE #2,3,3:P
RINT #2,"CARDS":LOCATE #2,3,4:PRINT #2,"FOR BOX"
260 LET C=6:GOSUB 2810
270 LET M=3
280 LET Z$=INKEY$:IF Z$="" THEN GOTO 280
290 LET Z=VAL(Z$):IF Z>6 OR Z<1 THEN GOTO 280
300 LET C$(M)=A$(Z+6)
310 IF C$(M)=C$(3) AND M=4 THEN GOTO 280
320 LOCATE #2,1+3*(M-2),6:PEN #2,I(Z):PRINT #2,R$(
Z):LOCATE #2,1+3*(M-2),7:PRINT #2,Q$(Z):PEN #2,1:L
ET M=M+1
330 IF M<>5 THEN GOTO 280
340 LET K=1
350 FOR I=7 TO 12
360 IF A$(I)<>C$(3) AND A$(I)<>C$(4) THEN LET P$(K
)=A$(I):LET K=K+1
370 NEXT I
380 CLS #4:PAPER 3:FOR I=1 TO C:LOCATE 3*(I-1)+5,2
5:PRINT " ";:NEXT I:PAPER 0
390 FOR I=1 TO 4:LET A$(I)=P$(I):NEXT I
400 LET C1=1:LET C2=4:LET CN=4:GOSUB 2960:GOSUB 30
70:GOSUB 3200:GOSUB 2600
410 GOSUB 2880:LOCATE #2,3,2:PRINT #2,"PRESS":LOCA
TE #2,3,3:PRINT #2,"ENTER":LOCATE #2,3,4:PRINT #2,
"TO CUT":LOCATE #2,3,5:PRINT #2,"THE":LOCATE #2,3,
6:PRINT #2,"CARDS"
420 IF INKEY$="" THEN GOTO 420
430 LET A=INT(RND(1)*4)+1
440 LET F$(1)=MID$(S$,A,1)
450 LET A=INT(RND(1)*26)+1
460 IF A/2=INT(A/2) THEN GOTO 450
470 LET F$(1)=F$(1)+MID$(V$,A,2)
480 FOR I=1 TO 12
490 IF B$(I)=F$(1) THEN GOTO 430
500 NEXT I
510 CLS #5
520 LET X1=486:LET Y1=230:GOSUB 3270
530 LET C1=1:LET C2=1:LET CN=1:LET A$(1)=F$(1):GOS
UB 3070:LET Y$=Q$(1):LET Z$=R$(1):LET II=32:GOSUB
2620:FOR ZZ=1 TO 2000:NEXT ZZ
540 IF RIGHT$(F$(1),2)="11" AND GO/2=INT(GO/2) THE
N LET SCOREC=SCOREC+2:GOSUB 1270
550 IF RIGHT$(F$(1),2)="11" AND GO/2<>INT(GO/2) TH
EN LET SCOREP=SCOREP+2:GOSUB 1270
560 GOSUB 2920:GOSUB 2880 :GOSUB 1450:GOSUB 2920:I
F GO/2<>INT(GO/2) THEN LET FL=1:GOTO 770
570 GOSUB 2920:GOSUB 2880:LET C1=1:LET C2=1:LET CN
=1:LET A$(1)=F$(1):GOSUB 3070:LET Y$=Q$(1):LET Z$=
R$(1):LET II=32:GOSUB 2620:LET X1=486:LET Y1=230:G
OSUB 3270
```

```
580 FOR I=1 TO 4:LET A$(I)=P$(I):NEXT I
590 LET C1=1:LET C2=4:LET CN=4
600 GOSUB 2960:GOSUB 3070:GOSUB 3200:GOSUB 2600
610 FOR I=1 TO 4
620 IF VAL(RIGHT$(F$,2))>VAL(RIGHT$(P$,2)) THEN NE
XT I
630 FOR J=5 TO I+1 STEP -1
640 LET P$(J)=P$(J-1)
650 NEXT J
660 LET P$(I)=F$(1)
670 LET I$(5)=F$(1)
680 LET C$(5)=F$(1)
690 GOSUB 2880
700 LOCATE #2,3,2:PRINT #2,"PLAYERS":LOCATE #2,4,3
:PRINT #2,"SCORE ";
710 FOR I=1 TO 5:LET H$(I)=P$(I):NEXT I:GOSUB 3360
720 LOCATE #2,5,5:PRINT #2,S
730 FOR ZZ=1 TO 2000:NEXT ZZ
740 LET SCOREP=SCOREP+S
750 GOSUB 1270
760 IF FL=1 THEN LET FL=0:GOTO 990
770 GOSUB 2880
780 MOVE 488,390:TAG:PRINT "COMPUTERS";:TAGOFF:LOC
ATE#2,3,3:PRINT#2,"SCORE ";
790 GOSUB 2920:LET C1=1:LET C2=1:LET CN=1:LET A$(1
)=F$(1):GOSUB 3070:LET Y$=Q$(1):LET Z$=R$(1):LET I
I=32:GOSUB 2620:LET X1=486:LET Y1=230:GOSUB 3270
800 FOR I=1 TO 4
810 LET A$(I)=I$(I)
820 NEXT I
830 LET C1=1:LET C2=4:LET CN=4
840 GOSUB 2960:GOSUB 3070:GOSUB 3200:GOSUB 2600
850 FOR I=1 TO 5:LET H$(I)=I$(I):NEXT I
860 FOR I=1 TO 4
870 IF VAL(RIGHT$(F$(1),2))>VAL(RIGHT$(H$(I),2)) T
HEN NEXT I
880 FOR J=5 TO I+1 STEP -1
890 LET H$(J)=H$(J-1)
900 NEXT J
910 LET H$(I)=F$(1)
920 GOSUB 3360
930 LOCATE #2,5,5:PRINT #2,S
940 FOR ZZ=1 TO 2000:NEXT ZZ
950 LET SCOREC=SCOREC+S
960 GOSUB 1270
970 GOSUB 2880
980 IF FL=1 THEN GOTO 570
990 GOSUB 2880:IF GO/2<>INT(GO/2) THEN LOCATE #2,2
,2:PRINT #2,"PLAYER'S"
1000 IF GO/2=INT(GO/2) THEN MOVE 488,390:TAG:PRINT
 "COMPUTERS";:TAGOFF
```

```
1010 LOCATE #2,2,3:PRINT #2,"CRIBBAGE":LOCATE #2,3
,4:PRINT #2,"SCORE ";
1020 CLS #4
1030 FOR I=1 TO 4
1040 LET A$(I)=C$(I)
1050 NEXT I
1060 LET C1=1:LET C2=4:LET CN=4
1070 GOSUB 2960:GOSUB 3070:GOSUB 3200: GOSUB 2600
1080 FOR I=1 TO 5
1090 LET B$(I)=C$(I)
1100 NEXT I
1110 LET S=1:LET F=5:GOSUB 4550
1120 FOR I=1 TO 5
1130 LET H$(I)=A$(I)
1140 NEXT I
1150 GOSUB 3360
1160 LOCATE #2,5,6:PRINT #2,S
1170 FOR ZZ=1 TO 2000:NEXT ZZ
1180 IF GO/2=INT(GO/2) THEN LET SCOREC=SCOREC+S:GO
TO 1200
1190 LET SCOREP=SCOREP+S
1200 GOSUB 1270
1210 LET GO=GO+1
1220 CLS#4:CLS #5
1230 GOSUB 2880
1240 GOTO 90
1250 IF INKEY$="" THEN GOTO 1250
1260 END
1270 REM ** SCORE BOARD **
1280 LET SP1=4+SCOREP*14
1290 LET SC1=4+SCOREC*14
1300 IF Z$<>"" THEN LET Z$=""
1310 IF SCOREC>=61 AND RC=1 THEN LET Z$="COMPUTER"
1320 IF SCOREP>=61 AND RP=1 THEN LET Z$="PLAYER"
1330 GOSUB 4720
1340 IF SCOREP>60 THEN LET SCOREP=SCOREP-60:LET RP
=1:LET SP1=4+14*SCOREP:LET SP2=295
1350 IF SCOREC>60 THEN LET SCOREC=SCOREC-60:LET RC
=1:LET SC1=4+14*SCOREC:LET SC2=375
1360 IF SCOREC>30 THEN LET SC2=360:LET SC1=424-(SC
OREC-31)*14
1370 IF SCOREP>30 THEN LET SP2=310:LET SP1=424-(SC
OREP-31)*14
1380 IF Z$="" THEN GOTO 1410
1390 IF Z$="COMPUTER" THEN LET SC1=18:LET SC2=335
1400 IF Z$="PLAYER" THEN LET SP1=18:LET SP2=335
1410 PLOT SC1,SC2,1:DRAW SC1-2,SC2+2:DRAW SC1+2,SC
2+2:DRAW SC1,SC2:DRAW SC1,SC2+10
1420 PLOT SP1,SP2,1:DRAW SP1-2,SP2+2:DRAW SP1+2,SP
2+2:DRAW SP1,SP2:DRAW SP1,SP2+10
1430 IF Z$<>"" THEN GOTO 4230
```

```
1440 RETURN
1450 REM ** PLAY ROUTINE **
1460 LET PO=GO+1:LET PASS=0:LET CP=0:LET TCP=0:LET
 TOTAL=0
1470 PEN #2,1
1480 FOR I=1 TO 4
1490 LET D$(I)=P$(I)
1500 LET E$(I)=I$(I)
1510 NEXT I
1520 LOCATE #2,2,2:PRINT #2,"PLAY FOR":LOCATE #2,3
,3:PRINT #2,"POINTS"
1530 LOCATE #2,3,5:PRINT #2,"PRESS":LOCATE #2,2,6:
PRINT #2,"ENTER TO":LOCATE #2,2,7:PRINT #2,"CONTIN
UE"
1540 IF INKEY$="" THEN GOTO 1540
1550 IF PO/2<>INT(PO/2) THEN GOSUB 1660
1560 IF TOTAL=31 THEN LET TOTAL=0:LET PASS=0:LET C
P=0:GOSUB 2920:GOTO 1620
1570 IF PASS=2 THEN LET SCOREP=SCOREP+1:GOSUB 1270
:GOSUB 2920:LET TOTAL=0:LET CP=0:LET PASS=0:GOTO 1
620
1580 IF TCP=8 THEN GOTO 1630
1590 IF PO/2=INT(PO/2) THEN GOSUB 2010
1600 IF TOTAL=31 THEN LET TOTAL=0:LET PASS=0:LET C
P=0:GOSUB 2920:GOTO 1620
1610 IF PASS=2 THEN LET SCOREC=SCOREC+1:GOSUB 1270
:GOSUB 2920:LET TOTAL=0:LET CP=0:LET PASS=0
1620 IF TCP<8 THEN GOTO 1550
1630 IF PO/2=INT(PO/2) THEN LET SCOREP=SCOREP+1:GO
SUB 1270
1640 IF PO/2<>INT(PO/2) THEN LET SCOREC=SCOREC+1:G
OSUB 1270
1650 RETURN
1660 REM ** PLAYERS GO **
1670 GOSUB 2880
1680 PEN #2,1:LOCATE #2,4,2:PRINT #2,"YOUR":LOCATE
 #2,4,3:PRINT #2,"TURN"
1690 FOR I=1 TO 4:LET A$(I)=D$(I):NEXT I:LET C1=1:
LET C2=4:GOSUB 3070
1700 FOR I=1 TO 4
1710 IF D$(I)="999" THEN GOTO 1780
1720 IF ASC(Q$(I))=227 OR ASC(Q$(I))=228 THEN PEN
#2,2
1730 LOCATE #2,1+2*I,5:PRINT #2,Q$(I):LOCATE #2,1+
2*I,6:PRINT #2,R$(I)
1740 PEN #2,3
1750 LOCATE #2,2*I,7:PRINT #2,I
1760 LOCATE #2,2*I,7:PRINT #2,I;
1770 PEN #2,1
1780 NEXT I
1790 GOSUB 2900
```

```
1800 LET Y$=INKEY$:IF Y$="" THEN GOTO 1800
1810 IF Y$="P" THEN LET PO=PO+1:LET PASS=PASS+1:RE
TURN
1820 IF ASC(Y$)>57 THEN GOTO 1800
1830 LET Y=VAL(Y$):IF Y>4 THEN GOTO 1800
1840 IF D$(Y)="999" THEN GOTO 1800
1850 LET VL=VAL(RIGHT$(A$(Y),2)):IF VL>10 THEN LET
 VL=10
1860 IF TOTAL+VL>31 THEN GOTO 1800
1870 LET TOTAL=TOTAL+VL
1880 LET P(CP+1)=VL
1890 LET PO=PO+1
1900 LET CP=CP+1
1910 LET TCP=TCP+1
1920 LET G$(CP)=D$(Y)
1930 LET D$(Y)="999"
1940 FOR I=1 TO CP
1950 LET A$(I)=G$(I)
1960 NEXT I
1970 GOSUB 2920
1980 LET C1=1:LET C2=CP:LET CN=CP:GOSUB 2960
1990 GOSUB 3070:GOSUB 3200:GOSUB 2600:GOSUB 2350
2000 RETURN
2010 REM ** COMPUTER'S GO **
2020 FOR I=1 TO 4:LET A$(I)=E$(I):NEXT I:LET C1=1:
LET C2=4:GOSUB 3070
2030 GOSUB 2880
2040 MOVE 488,390:TAG:PRINT "COMPUTERS";:TAGOFF:LO
CATE#2,4,3:PRINT#2,"PLAY"
2050 LOCATE #2,3,6:PRINT #2,"PLEASE"
2060 LOCATE #2,4,7:PRINT #2,"WAIT"
2070 FOR Y=1 TO 4
2080 LET VL=VAL(RIGHT$(E$(Y),2))
2090 IF VL=99 THEN GOTO 2130
2100 IF VL>10 THEN LET VL=10
2110 IF TOTAL+VL=15 OR TOTAL+VL=31 THEN GOTO 2210
2120 IF CP>=1 THEN IF VL=P(CP) AND TOTAL+VL<=31 TH
EN GOTO 2210
2130 NEXT Y
2140 FOR Y=4 TO 1 STEP -1
2150 IF E$(Y)="999" THEN GOTO 2190
2160 LET VL=VAL(RIGHT$(E$(Y),2))
2170 IF VL>10 THEN LET VL=10
2180 IF TOTAL+VL<=31 THEN GOTO 2210
2190 NEXT Y
2200 LET PO=PO+1:LET PASS=PASS+1:RETURN
2210 REM ** PLAY CARD **
2220 LET P(CP+1)=VL
2230 LET CP=CP+1
2240 LET G$(CP)=E$(Y)
2250 LET E$(Y)="999"
```

```
2260 LET TCP=TCP+1:LET TOTAL=TOTAL+VL
2270 LET PO=PO+1
2280 FOR I=1 TO CP
2290 LET A$(I)=G$(I)
2300 NEXT I
2310 GOSUB 2920
2320 LET C1=1:LET C2=CP:LET CN=CP
2330 GOSUB 2960:GOSUB 3070:GOSUB 3200:GOSUB 2600:G
OSUB 2350
2340 RETURN
2350 REM ** SCORE EVALUATION **
2360 LET SPG=0:LET TT=0:LET PASS=0
2370 IF TOTAL=15 OR TOTAL=31 THEN  LET SPG=2
2380 FOR I=1 TO CP
2390 LET P(I)=VAL(RIGHT$(G$(I),2))
2400 NEXT I
2410 IF CP<=2 THEN GOTO 2540
2420 LET MX=P(CP):LET MN=P(CP)
2430 FOR I=1 TO CP-2
2440 LET TT=0:LET MX=P(CP):LET MN=P(CP)
2450 FOR J=CP TO I STEP -1
2460 IF P(J)>MX THEN LET MX=P(J)
2470 IF P(J)<MN THEN LET MN=P(J)
2480 LET TT=TT+P(J)
2490 NEXT J
2500 LET AV1=(MX+MN)/2
2510 LET AV2=TT/((CP-I)+1)
2520 IF AV1=AV2 AND MX-MN=CP-I THEN LET SPG=SPG+CP
+1-I:GOTO 2540
2530 NEXT I
2540 IF CP>=3 THEN IF P(CP)=P(CP-1) AND P(CP-1)=P(
CP-2) THEN LET SPG=SPG+6:GOTO 2560
2550 IF CP>=2 THEN IF P(CP)=P(CP-1) THEN LET SPG=S
PG+2
2560 IF PO/2=INT(PO/2) THEN LET SCOREP=SCOREP+SPG
2570 IF PO/2<>INT(PO/2) THEN LET SCOREC=SCOREC+SPG
2580 GOSUB 1270
2590 RETURN
2600 REM ** SINGLE CARD **
2610 LET JJ=13:LET II=(3*CN)+1:LET Y$=Q$(C2):LET Z
$=R$(C2)
2620 IF ASC(Y$)=227 OR ASC(Y$)=228 THEN PEN 2
2630 IF ASC(Y$)=226 OR ASC(Y$)=229 THEN PEN 1
2640 LOCATE II,JJ:PRINT Y$
2650 LOCATE II,JJ-1:PRINT Z$
2660 LOCATE II+6,JJ+8:PRINT Y$
2670 LOCATE II+6,JJ+9:PRINT Z$
2680 IF Z$="J" THEN LOCATE II+2,JJ+3:PRINT CHR$(13
0);CHR$(139);CHR$(131):LOCATE II+3,JJ+4:PRINT CHR$
(138):LOCATE II+2,JJ+5:PRINT CHR$(137);CHR$(134)
2690 IF Z$="Q" THEN LOCATE II+2,JJ+3:PRINT CHR$(13
```

```
4);CHR$(131);CHR$(137):LOCATE II+2,JJ+4:PRINT CHR$
(133);CHR$(136);CHR$(138):LOCATE II+2,JJ+5:PRINT C
HR$(137);CHR$(140);CHR$(135):LOCATE II+4,JJ+6:PRIN
T CHR$(130)
2700 IF Z$="K" THEN LOCATE II+2,JJ+3:PRINT CHR$(13
3);CHR$(136);CHR$(129):LOCATE II+2,JJ+4:PRINT CHR$
(135);CHR$(137):LOCATE II+2,JJ+5:PRINT CHR$(133);"
 ";CHR$(137)
2710 IF ASC(Z$)=240 OR Z$="9" OR Z$="8" THEN FOR J
1=JJ+1 TO JJ+7 STEP 2:LOCATE II+2,J1:PRINT Y$;" ";
Y$:NEXT J1
2720 IF ASC(Z$)=240 THEN LOCATE II+3,JJ+2:PRINT Y$
:LOCATE II+3,JJ+6:PRINT Y$
2730 IF Z$="9" THEN LOCATE II+3,JJ+4:PRINT Y$
2740 IF Z$="6" OR Z$="7" THEN FOR J1=JJ+1 TO JJ+7
STEP 3:LOCATE II+2,J1:PRINT Y$;" ";Y$:NEXT J1
2750 IF Z$="7" THEN LOCATE II+3,JJ+4:PRINT Y$
2760 IF Z$="5" OR Z$="4" THEN LOCATE II+2,JJ+2:PRI
NT Y$;" ";Y$:LOCATE II+2,JJ+6:PRINT Y$;" ";Y$
2770 IF Z$="5" THEN LOCATE II+3,JJ+4:PRINT Y$
2780 IF Z$="3" OR Z$="2" THEN LOCATE II+3,JJ+2:PRI
NT Y$:LOCATE II+3,JJ+6:PRINT Y$
2790 IF Z$="3" OR Z$="1" THEN LOCATE II+3,JJ+4:PRI
NT Y$
2800 RETURN
2810 REM ** NUMBER CARDS **
2820 FOR I=1 TO C
2830 PAPER 3:PEN 1
2840 LOCATE 3*(I-1)+4,25:PRINT I;
2850 NEXT I
2860 PAPER 0:PEN 1
2870 RETURN
2880 REM ** CLEAR REPORT **
2890 CLS #2
2900 PLOT 480,399,1:DRAW 639,399:DRAW 639,272:DRAW
 480,272:DRAW 480,399
2910 RETURN
2920 REM ** CLEAR BOTTOM **
2930 CLS #3
2940 PLOT 1,30,1:DRAW 1,240:DRAW 639,240:DRAW 639,
30:DRAW 1,30
2950 RETURN
2960 REM ** DRAW CARDS **
2970 FOR I=165 TO 165+48*CN-3 STEP 48
2980 PLOT I,230,1
2990 DRAW I-122,230
3000 DRAW I-125,227
3010 DRAW I-125,48
3020 DRAW I-122,44
3030 DRAW I,44
3040 NEXT I
```

```
3050 DRAWR 3,3:DRAWR 0,180:DRAWR -3,3
3060 RETURN
3070 REM** PLAYER'S CARDS **
3080 FOR I=C1 TO C2
3090 IF LEFT$(A$(I),1)="H" THEN LET Q$(I)=CHR$(228
):LET I(I)=2
3100 IF LEFT$(A$(I),1)="C" THEN LET Q$(I)=CHR$(226
):LET I(I)=1
3110 IF LEFT$(A$(I),1)="D" THEN LET Q$(I)=CHR$(227
):LET I(I)=2
3120 IF LEFT$(A$(I),1)="S" THEN LET Q$(I)=CHR$(229
):LET I(I)=1
3130 IF MID$(A$(I),2,1)="0" THEN LET R$(I)=RIGHT$(
A$(I),1):GOTO 3180
3140 IF RIGHT$(A$(I),1)="0" THEN LET R$(I)=CHR$(24
0)
3150 IF RIGHT$(A$(I),1)="1" THEN LET R$(I)="J"
3160 IF RIGHT$(A$(I),1)="2" THEN LET R$(I)="Q"
3170 IF RIGHT$(A$(I),1)="3" THEN LET R$(I)="K"
3180 NEXT I
3190 RETURN
3200 REM ** COMPLETE CARDS **
3210 FOR I=1 TO CN
3220 PEN I(I)
3230 LOCATE 3*(I-1)+4,12:PRINT R$(I)
3240 LOCATE 3*(I-1)+4,13:PRINT Q$(I)
3250 NEXT I
3260 RETURN
3270 REM ** CARD OUTLINE **
3280 PLOT X1,Y1,1:DRAWR 0,-180:DRAWR 3,-3:DRAWR 12
4,0:DRAWR 3,3:DRAWR 0,180:DRAWR -3,3:DRAWR -124,0:
DRAWR -3,-3
3290 RETURN
3300 REM ** SHADE PACK
3310 PEN 2
3320 LOCATE 31,11:PRINT CHR$(241);STRING$(7,CHR$(2
18));CHR$(242)
3330 FOR J=12 TO 22:LOCATE 31,J:PRINT CHR$(217);ST
RING$(7,CHR$(207));CHR$(219):NEXT J
3340 PEN 1
3350 RETURN
3360 REM ** EVALUATER **
3370 LET S=0
3380 LET SP=0
3390 FOR X=1 TO 5
3400 LET C(X)=VAL(RIGHT$(H$(X),2))
3410 NEXT X
3420 GOTO 3830
3430 REM ** BONUS **
3440 FOR X=1 TO 5
3450 IF H$(X)<>F$(1) AND RIGHT$(H$(X),2)="11" AND
```

```
LEFT$(H$(X),1)=LEFT$(F$(1),1) THEN LET S=S+1
3460 IF C(X)>10 AND C(X)<14 THEN LET C(X)=10
3470 NEXT X
3480 REM ** PAIRS **
3490 FOR X=1 TO 4
3500 FOR Y=X+1 TO 5
3510 IF RIGHT$(H$(X),2)=RIGHT$(H$(Y),2) THEN LET S
P=SP+2
3520 NEXT Y:NEXT X
3530 LET S=S+SP
3540 REM ** FLUSH **
3550 LET SF=0
3560 IF LEFT$(H$(1),1)<>LEFT$(F$(1),1) THEN LET Y$
=LEFT$(H$(1),1):LET XX=1
3570 IF LEFT$(H$(1),1)=LEFT$(F$(1),1) THEN LET Y$=
LEFT$(H$(2),1):LET XX=0
3580 FOR Z=2 TO 5
3590 IF LEFT$(H$(Z),1)=Y$ AND LEFT$(H$(Z),1)<>LEFT
$(F$(1),1) THEN LET XX=XX+1
3600 NEXT Z
3610 IF XX=4 THEN LET SF=4:GOTO 3630
3620 IF LEFT$(H$(1),1)=LEFT$(H$(2),1) AND LEFT$(H$
(1),1)=LEFT$(H$(3),1) AND LEFT$(H$(1),1)=LEFT$(H$(
4),1) AND LEFT$(H$(1),1)=LEFT$(H$(5),1) THEN LET S
=S+5
3630 LET S=S+SF
3640 REM ** 2-CARD 15 **
3650 FOR X=1 TO 4
3660 FOR Y=X+1 TO 5
3670 IF C(X)+C(Y)=15 THEN LET S=S+2
3680 NEXT Y
3690 NEXT X
3700 REM ** 3-CARD 15 **
3710 FOR X=1 TO 3
3720 FOR Y=X+1 TO 4
3730 FOR Z=Y+1 TO 5
3740 IF C(X)+C(Y)+C(Z)=15 THEN LET S=S+2
3750 NEXT Z:NEXT Y:NEXT X
3760 IF C(1)+C(2)+C(3)+C(4)+C(5)=15 THEN LET S=S+2
3770 IF C(1)+C(2)+C(3)+C(4)=15 THEN LET S=S+2
3780 IF C(1)+C(2)+C(3)+C(5)=15 THEN LET S=S+2
3790 IF C(1)+C(2)+C(4)+C(5)=15 THEN LET S=S+2
3800 IF C(1)+C(3)+C(4)+C(5)=15 THEN LET S=S+2
3810 IF C(2)+C(3)+C(4)+C(5)=15 THEN LET S=S+2
3820 GOTO 3980
3830 IF C(1)+1=C(2) AND C(2)+1=C(3) AND C(3)+1=C(4
) AND C(4)+1=C(5) THEN LET S=S+5:GOTO 3970
3840 LET R4=0
3850 FOR W=1 TO 2
3860 FOR X=W+1 TO 3
3870 FOR Y=X+1 TO 4
```

```
3880 FOR Z=Y+1 TO 5
3890 IF C(W)=C(X)-1 AND C(X)=C(Y)-1 AND C(Y)=C(Z)-
1 THEN LET R4=R4+4
3900 NEXT Z:NEXT Y:NEXT X:NEXT W
3910 IF R4<>0 THEN LET S=S+R4:GOTO 3970
3920 FOR X=1 TO 3
3930 FOR Y=X+1 TO 4
3940 FOR Z=Y+1 TO 5
3950 IF C(Y)-C(X)=1 AND C(Z)-C(Y)=1 THEN LET S=S+3
3960 NEXT Z:NEXT Y:NEXT X
3970 GOTO 3430
3980 IF S>MX THEN LET MX=S:LET N1=I:LET N2=J:LET N
3=K:LET N4=L
3990 RETURN
4000 REM ** HAND SELECTION **
4010 LET H$(5)="999"
4020 LET MX=0
4030 FOR I=1 TO 3
4040 FOR J=I+1 TO 4
4050 FOR K=J+1 TO 5
4060 FOR L=K+1 TO 6
4070 LET H$(1)=A$(I)
4080 LET H$(2)=A$(J)
4090 LET H$(3)=A$(K)
4100 LET H$(4)=A$(L)
4110 GOSUB 3360
4120 NEXT L:NEXT K:NEXT J:NEXT I
4130 IF MX=0 THEN LET N1=1:LET N2=2:LET N3=3:LET N
4=4
4140 LET I$(1)=A$(N1):LET I$(2)=A$(N2):LET I$(3)=A
$(N3):LET I$(4)=A$(N4)
4150 LET K=1
4160 FOR I=1 TO 6
4170 FOR J=1 TO 4
4180 IF A$(I)=I$(J) THEN GOTO 4210
4190 NEXT J
4200 LET C$(K)=A$(I):LET K=K+1
4210 NEXT I
4220 RETURN
4230 REM ** GAME OVER **
4240 FOR I=1 TO 4
4250 FOR J=1 TO 23
4260 FOR ZZ=1 TO 50:NEXT ZZ
4270 BORDER J
4280 NEXT J:NEXT I
4290 BORDER 18
4300 GOSUB 2880
4310 LOCATE #2,4,2:PRINT #2,"GAME":LOCATE #2,4,3:P
RINT #2,"OVER"
4320 IF Z$="COMPUTER" THEN LOCATE #2,2,4:PEN #2,2:
PRINT #2,Z$:PEN #2,1:LOCATE #2,4,5:PRINT #2,"WINS"
```

```
4330 IF Z$="PLAYER" THEN LOCATE #2,3,4:PEN #2,2:PR
INT #2,Z$:PEN #2,1:LOCATE #2,4,5:PRINT #2,"WINS"
4340 LOCATE #2,2,7:PRINT #2,"ENTER TO":LOCATE #2,2
,8:PRINT #2,"CONTINUE";
4350 GOSUB 2900
4360 IF INKEY$="" THEN GOTO 4360
4370 RUN
4380 REM **CARDS**
4390 LET S$="HCDS"
4400 LET V$="0102030405060708091011121 3"
4410 FOR I=1 TO 12
4420 LET A=INT(RND(1)*4)+1
4430 LET A$(I)=MID$(S$,A,1)
4440 LET A=INT(RND(1)*26)+1
4450 IF A/2=INT(A/2) THEN GOTO 4440
4460 LET A$(I)=A$(I)+MID$(V$,A,2)
4470 FOR J=1 TO I-1
4480 IF A$(J)=A$(I) THEN GOTO 4420
4490 NEXT J
4500 NEXT I
4510 RETURN
4520 FOR I=S TO F
4530 LET B$(I)=A$(I)
4540 NEXT I
4550 FOR J=S TO F
4560 LET MN=99
4570 FOR I=S TO F
4580 IF VAL(RIGHT$(B$(I),2))<MN THEN GOSUB 4640
4590 NEXT I
4600 LET A$(J)=B$(K)
4610 LET B$(K)=LEFT$(B$(K),1)+"99"
4620 NEXT J
4630 RETURN
4640 LET MN=VAL(RIGHT$(B$(I),2))
4650 LET K=I
4660 RETURN
4670 REM **TEMPORARY TRANSFER**
4680 FOR I=1 TO 12
4690 LET B$(I)=A$(I)
4700 NEXT I
4710 RETURN
4720 REM ** CRIB BOARD **
4730 CLS #1:PLOT 1,399,1:DRAW 446,399:DRAW 446,272
:DRAW 1,272:DRAW 1,399
4740 FOR I=1 TO 30
4750 PLOT 4+I*14,375,1:PLOT 4+I*14,360:PLOT 4+I*14
,295,1:PLOT 4+I*14,310
4760 NEXT I
4770 PLOT 18,335,1
4780 RETURN
4790 REM ** WINDOWS **
```

```
4800 WINDOW #1,1,28,1,8:PAPER #1,0:PEN #1,1
4810 WINDOW #2,31,40,1,8:PAPER #2,0:CLS #2:PEN #2,
1
4820 PLOT 480,399,1:DRAW 639,399:DRAW 639,272:DRAW
 480,272:DRAW 480,399
4830 WINDOW #3,1,40,11,23:PAPER #3,0:CLS #3
4840 PLOT 1,30,1:DRAW 1,240:DRAW 639,240:DRAW 639,
30:DRAW 1,30
4850 WINDOW #4,2,28,11,23:PAPER #4,0
4860 WINDOW #5,30,39,11,23:PAPER #5,0
4870 SYMBOL AFTER 240
4880 SYMBOL 240,&66,&E9,&69,&69,&69,&69,&66,&0
4890 SYMBOL 241,&0,&0,&0,&0,&A,&5,&A,&5
4900 SYMBOL 242,&0,&0,&0,&0,&80,&50,&A0,&50
4910 RETURN
```

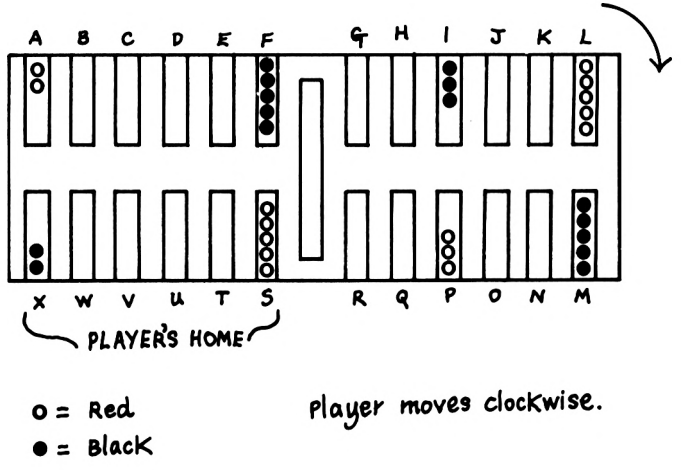
10. Backgammon

This is your chance to undertake one of the ultimate boardgame challenges as the Amstrad offers you a game of Backgammon. The game, which is displayed in full graphic detail with prompting messages to direct you in your moves, contains all the standard features of Backgammon and also includes the doubling die to add extra excitement.

This version assumes a basic knowledge of the game. If you are uncertain of the rules then we recommend that you first read about Backgammon to acquire the necessary understanding before accepting the Amstrad's challenge.

Playing instructions

When the program is executed, the board, dice and doubling die are displayed on the screen, with all the pieces positioned in their initial places. You have the red pieces and are moving clockwise around the board; the computer has the black pieces and is moving in the counter-clockwise direction.



The game begins when the dice are rolled to decide who is to have the first move and the initial possession of the doubling die. When it is the computer's turn, it displays a message while it is deciding which of its counters to reposition. When this decision has been made it will display a message similar to

DIE 1 = X

which means that the computer is moving a character positioned in column X by the amount shown on die 1.

For your turn, you must first roll the dice by pressing the ENTER key and then you are asked if you wish to change the order of the dice. This is because you always move by the amount shown on the left-hand die (DIE 1) first, so be certain that the dice are in the correct order before you proceed. Then you move your counters by pressing the key of the letter which corresponds to the column you wish to move from. If the move is illegal then the computer will not accept it and you must try again. Should there be no legal move at any stage then you can pass by typing Z.

The bar in the centre of the board shows the pieces which have been removed from play, and these must be replaced before any further moves can be made.

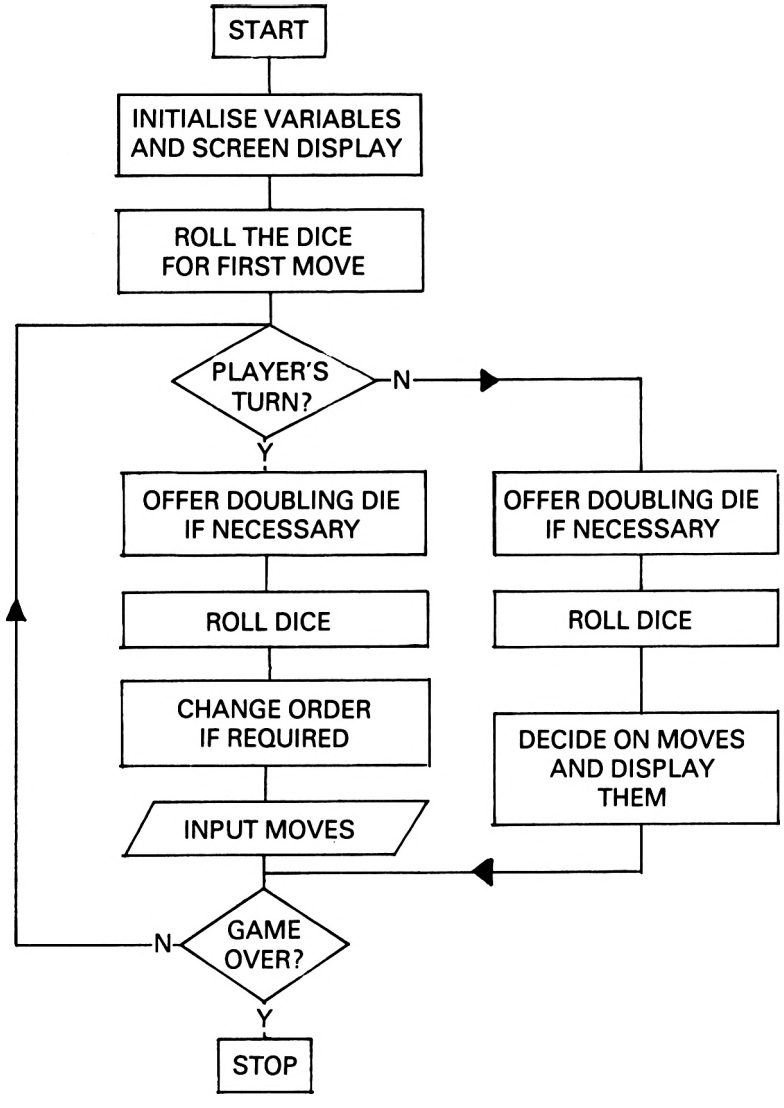
As with a normal Backgammon board, there is a limit to the number of pieces which can be positioned in any column before they are stacked on top of each other, and this limit is six. Should more than six counters be positioned in any column, only six will be displayed.

The program

The artificial intelligence sections of this program use the methods of 'look-ahead' and 'weighting' to decide on the best possible move for the computer. These routines take between thirty and ninety seconds depending on the number of legal possibilities. The reason for these fairly long pauses is that the Amstrad is unable to remember the position from its last move and has to start from scratch each time.

Flowchart

The flowchart on p.166 represents the general operation of the program.



Variables

The following table represents the major variables used in this program and should help you to understand how the program operates.

B()	Current board position
B1()	Temporary board storage
B2()	
C()	Position of computer's pieces
P()	Position of player's pieces
M()	The scores on the dice
G1	Stores who is to have the next move
STATUS	Which player has the doubling die
TST	Decides if all pieces are home
A$	Letter of the column to be moved from
A	Corresponding number to A$
MX	Highest weighting
POB	Number of player's pieces off the board
COB	Number of computer's pieces off the board
PB	Number of player's pieces on the bar
CB	Number of computer's pieces on the bar
LEGAL	Holds the value 1 if the move is legal
MOVES	Number of moves to be taken (2 or 4)
PTS	Number on the doubling die

Comments on the program

The program is highly structured, and the following notes break it down into its various subroutines.

Lines 10-170
This section of the program is concerned with the initialisation of the variables, the formation of the user-defined graphics and the construction of the initial screen display, using some of the subroutines described below.

Lines 180-460
This is the main program, which handles the roll of the dice to decide on the player with the first move and then calls the 'player's turn' and 'computer's turn' routines in the appropriate order until the game is finished.

Lines 470-930
This is the routine which allows the user to roll the dice and move his counters if he has not had any placed on the bar by the computer.

Lines 940-1120
This section of code is related to lines 470-930, and is concerned with removing the player's pieces from the bar.

Lines 1130-1620
This major routine considers the movement of two separate computer counters and calls subroutines to consider other alternatives. The chosen move is then executed and displayed on the screen.

Lines 1630-1930
This routine considers moving the same computer counter twice.

Lines 1940-2090
A section of program which is called if the computer cannot find two legal moves. It decides if one move can be made or if there is no possible move and passes at the relevant position.

Lines 2100-2680
A routine which decides on the best position to occupy when a computer counter is to be replaced on the board from the bar.

Lines 2690-3020
The 'end game' section of code which is called when all the computer's counters are in its home and the time has come to remove them from the board.

Lines 3030-3260
This is where the computer decides to offer the doubling die if it is free to do so.

Lines 3270-3340
The routine which displays the winner and the winning margin at the end of the game.

Lines 3350-3510
This part of the program re-draws the counters which are positioned on the bar.

Lines 3520-3540
A short subroutine which switches the dice around.

Lines 3550-3570
This displays the 'Amstrad thinking' message which is shown while the computer decides on its move.

Lines 3580-3600
A section of code to remove the above message when the computer's move has been chosen.

Lines 3610-3690
This routine clears the spots from the dice.

Lines 3700-3770
This routine chooses random values in the range 1-6 for the values of the two dice.

Lines 3780-3870
The part of the program that places the spots in the appropriate positions to form the graphical representation of the two dice.

Lines 3880-3950
A short subroutine which is used to print the numerical value of the doubling die.

Lines 3960-4500
The subroutine which positions the counters in their correct places on the board.

Lines 4510-4680
The routine which is used to determine the legality of a move.

Lines 4690-4880
This subroutine calculates the weighting of the move currently under consideration.

Lines 4890-5020
A section of code called early in the program which assigns the initial values to the arrays B(), C() and P().

Lines 5030-5340
This subroutine constructs the original screen display.

Lines 5350-5390
This section of code creates the user-defined graphics which are to be used for the counters.

The listing

The listing should be entered very carefully and then saved on to cassette as outlined in the Introduction.

The game should be entered and played in the caps lock mode. All spaces within the text should be entered as they appear.

```
10 MODE 1:RANDOMIZE TIME
20 INK 0,26:INK 1,0:INK 2,6:INK 3,18
30 PAPER 0:PEN 1:CLS
40 BORDER 18
50 DIM M(4),C(15),P(15),C2(15),B(24),B1(24),B2(24)
60 LET MOVES=2:LET PTS=1:LET MM=0:LET Z$="":LET FL
=0:LET POB=0:LET COB=0:LET FL1=0
70 GOSUB 5350
80 WINDOW #1,3,29,12,23:PAPER #1,3
90 WINDOW #2,16,16,13,22:PAPER #2,0:CLS #2
100 WINDOW #3,18,38,2,9:PAPER #3,3:CLS #3
110 WINDOW #4,5,13,4,8:PAPER #4,3:CLS #4
120 WINDOW #5,33,37,16,20:PAPER #5,3:CLS #5
130 GOSUB 5030
140 GOSUB 4890
150 GOSUB 3960
160 GOSUB 3610
170 GOSUB 3880
180 REM ** MAIN PROGRAM **
190 LET G1=0
200 CLS #3:PRINT #3
210 PRINT #3,"    ROLL THE DICE"
220 PRINT #3,"   TO SEE WHO GOES"
230 PRINT #3,"        FIRST"
240 PRINT #3:PRINT #3:PRINT #3,"     PRESS ENTER"
250 PEN #4,2
260 IF INKEY$="" THEN GOTO 260
270 GOSUB 3610
280 CLS #3
290 LET YS=M(1)+M(2)
300 LOCATE #3,5,3:PRINT #3,"YOUR SCORE   ";YS
310 LOCATE #3,2,5:PRINT #3,"AMSTRAD'S SCORE ";
320 PEN #4,1
330 FOR Z=1 TO 1000:NEXT Z
340 GOSUB 3610
350 PRINT #3,M(1)+M(2)
360 IF M(1)+M(2)>YS THEN LET G1=1
370 IF M(1)+M(2)=YS THEN FOR Z=1 TO 1000:NEXT Z:GO
TO 180
380 LET STATUS =G1
390 FOR Z=1 TO 1000:NEXT Z
400 IF G1/2=INT(G1/2) THEN PEN #4,2:GOSUB 470
410 IF POB=15 THEN LET Z$="PLAYER":GOTO 3270
420 IF G1/2<>INT(G1/2) THEN PEN #4,1:GOSUB 1130
430 FOR Z=1 TO 1000:NEXT Z
440 IF COB=15 THEN LET Z$="COMPUTER":GOTO 3270
450 LET G1=G1+1
460 GOTO 390
470 REM ** PLAYER'S TURN **
480 IF STATUS/2<>INT(STATUS/2) OR PTS=64 THEN GOTO
 520
```

```
490 CLS #3:PRINT #3:PRINT #3,"    PLAYER'S TURN":P
RINT #3:PRINT #3:PRINT #3,"      OFFER THE":PRINT
#3," DOUBLING DIE (Y/N)?"
500 LET QQ$=INKEY$:IF QQ$="" THEN GOTO 500
510 IF QQ$="Y" THEN LET PTS=PTS*2:LET STATUS=STATU
S+1:PRINT #3:PRINT#3,"   AMSTRAD ACCEPTS":GOSUB 38
80:FOR ZZ=1 TO 1000:NEXT ZZ
520 LET Z$=""
530 CLS #3:PRINT #3:PRINT #3,"    PLAYER'S TURN"
540 PRINT #3:PRINT #3:PRINT #3,"  PRESS  'ENTER' T
O":PRINT #3,"    ROLL THE DICE"
550 IF INKEY$="" THEN GOTO 550
560 GOSUB 3610
570 CLS #3:PRINT #3:PRINT #3,"    PLAYER'S TURN":P
RINT #3:PRINT #3:PRINT #3,"  TYPE 'Z'  TO PASS":PR
INT #3,"    OR 'ENTER' TO":PRINT #3,"      CONTINU
E"
580 LET ZZ$=INKEY$:IF ZZ$="" THEN GOTO 580
590 IF ZZ$="Z" THEN RETURN
600 CLS #3:PRINT #3:PRINT #3,"    PLAYER'S TURN"
610 IF M(1)=M(2) THEN GOTO 670
620 PRINT #3:PRINT #3:PRINT #3," CHANGE ORDER (Y/N
)?"
630 LET Z$=INKEY$:IF Z$="" THEN GOTO 630
640 IF Z$<>"Y" THEN GOTO 670
650 GOSUB 3520
660 GOSUB 3610:GOTO 600
670 CLS #3:PRINT #3:PRINT #3,"    PLAYER'S TURN":P
RINT #3
680 FOR I=1 TO MOVES
690 LET TST=0:FOR II=19 TO 24
700 IF B(II)>0 THEN LET TST=TST+B(II)
710 NEXT II
720 LET TST=TST+POB
730 IF PB>0 THEN GOTO 940
740 PRINT #3,"      DIE";I;"= ";
750 LET A$=INKEY$:IF A$="" THEN GOTO 750
760 IF A$="Z" THEN RETURN
770 LET A=ASC(A$)-64
780 IF A<1 OR A>24 THEN GOTO 750
790 GOSUB 4510
800 IF LEGAL=0 THEN GOTO 750
810 PRINT #3,A$
820 IF TST=15 AND A+M(I)>24 THEN LET B(A)=B(A)-1:L
ET POB=POB+1:GOTO 900
830 LET B(A)=B(A)-1
840 IF B(A+M(I))<>-1 THEN GOTO 890
850 FOR II=1 TO 15
860 IF C(II)=A+M(I) THEN LET C(II)=25:GOTO 880
870 NEXT II
880 IF B(A+M(I))=-1 THEN LET B(A+M(I))=1:LET CB=CB
```

```
+1:GOTO 900
890 LET B(A+M(I))=B(A+M(I))+1
900 GOSUB 3960
910 IF POB=15 THEN RETURN
920 NEXT I
930 RETURN
940 REM ** PLAYER'S MOVE OFF BAR **
950 PRINT #3:PRINT #3,"    MOVE OFF BAR"
960 PRINT #3," PRESS ENTER TO MOVE"
970 LET DN$=INKEY$:IF DN$="" THEN GOTO 970
980 IF DN$="Z" THEN RETURN
990 IF B(M(I))>-2 THEN GOTO 1010
1000 LOCATE #3,5,7:PRINT #3,"ILLEGAL MOVE":FOR Z=1
 TO 1000:NEXT Z:LOCATE #3,5,7:PRINT #3,"
  ":GOTO 970
1010 LET PB=PB-1
1020 IF B(M(I))=-1 THEN LET B(M(I))=0:LET CB=CB+1
1030 LET B(M(I))=B(M(I))+1
1040 GOSUB 3960
1050 FOR II=1 TO 15
1060 IF C(II)=M(I) THEN LET C(II)=25
1070 NEXT II
1080 FOR II=1 TO 15
1090 IF P(II)=15 THEN LET P(II)=M(I):GOTO 1110
1100 NEXT II
1110 CLS #3:PRINT #3:PRINT #3,"    PLAYER'S TURN":
PRINT #3
1120 GOTO 920
1130 REM ** COMPUTER'S TURN **
1140 IF STATUS/2<>INT(STATUS/2) AND PTS <>64 THEN
GOSUB 3030
1150 CLS #3:PRINT #3:PRINT #3,"   COMPUTER'S TURN"
1160 GOSUB 3610
1170 LET MVS=0
1180 GOSUB 3550
1190 LET TST=0
1200 FOR II=1 TO 6
1210 IF B(II)<0 THEN LET TST=TST-B(II)
1220 NEXT II
1230 LET TST=TST+COB
1240 IF TST=15 THEN GOTO 2690
1250 LET MX=-20
1260 IF CB>0 THEN GOTO 2100
1270 FOR J=1 TO 15
1280 IF J>1 THEN IF C(J)=C(J-1) THEN GOTO 1420
1290 FOR K=1 TO 15
1300 IF J=K THEN GOTO 1410
1310 LET A=C(K)
1320 LET I=1
1330 GOSUB 4510
1340 IF LEGAL=0 THEN GOTO 1410
```

```
1350 LET A=C(J)
1360 LET I=2
1370 GOSUB 4510
1380 IF LEGAL=0 THEN GOTO 1410
1390 GOSUB 4690
1400 IF W>MX THEN LET MX=W:LET MV1=K:LET MV2=J
1410 NEXT K
1420 NEXT J
1430 LET SW=1
1440 GOSUB 1630
1450 IF SW=1 THEN LET SW=0:GOSUB 3520:GOTO 1440
1460 GOSUB 3520
1470 IF SW1=0 THEN GOSUB 3520:LET Z$="Y":GOSUB 361
0:LET Z$=""
1480 IF MX=-20 THEN GOTO 1940
1490 GOSUB 3580
1500 IF B(C(MV1)-M(1))=1 THEN LET B(C(MV1)-M(1))=0
:LET PB=PB+1:FOR I=1 TO 15:IF P(I)=C(MV1)-M(1) THE
N LET P(I)=25:NEXT I
1510 LET B(C(MV1))=B(C(MV1))+1:LET B(C(MV1)-M(1))=
B(C(MV1)-M(1))-1
1520 IF MVS=1 THEN LOCATE #3,7,6:PRINT #3,"DIE 3 =
 ";CHR$(64+C(MV1)):GOTO 1540
1530 LOCATE #3,7,4:PRINT #3,"DIE 1 = ";CHR$(64+C(M
V1))
1540 LET C(MV1)=C(MV1)-M(1):GOSUB 3960
1550 IF B(C(MV2)-M(2))=1 THEN LET B(C(MV2)-M(2))=0
:LET PB=PB+1
1560 LET B(C(MV2))=B(C(MV2))+1:LET B(C(MV2)-M(2))=
B(C(MV2)-M(2))-1
1570 IF MVS=1 THEN LOCATE #3,7,7:PRINT #3,"DIE 4 =
 ";CHR$(64+C(MV2)):LET MVS=0:GOTO 1590
1580 LOCATE #3,7,5:PRINT #3,"DIE 2 = ";CHR$(64+C(M
V2))
1590 GOSUB 3960
1600 LET C(MV2)=C(MV2)-M(2)
1610 IF MOVES=4 THEN LET MVS=1:LET MOVES=2:GOTO 11
80
1620 RETURN
1630 REM ** COMPUTER'S ONE-PIECE MOVE **
1640 LET SW1=-3
1650 FOR I=1 TO 24:LET B2(I)=B(I):NEXT I
1660 FOR J=1 TO 15
1670 LET WW=0
1680 FOR I=1 TO 15
1690 LET C2(I)=C(I)
1700 NEXT I
1710 IF FL<>1 THEN FOR K=1 TO 2
1720 LET A=C2(J):LET I=K
1730 GOSUB 4510
1740 IF LEGAL=0 THEN GOTO 1880
```

```
1750 IF B(C2(J)-M(K))=1 THEN LET WW=WW+4
1760 IF B(C2(J)-M(K))=1 THEN LET B(C2(J)-M(K))=0
1770 LET B(C2(J)-M(K))=B(C2(J)-M(K))-1
1780 LET B(C2(J))=B(C2(J))+1
1790 LET C2(J)=C2(J)-M(K)
1800 IF FL=0 THEN NEXT K
1810 IF TST>11 AND B2(J)>7 THEN LET WW=WW+10
1820 FOR I=1 TO 24
1830 IF B(I)=-1 THEN LET WW=WW-3
1840 IF I<=6 AND B(I)<0 THEN LET WW=WW+1
1850 IF B(I)<0 AND I>18 THEN LET WW=WW-(I-18)*ABS(
B(I))
1860 NEXT I
1870 IF WW>MX THEN LET MX=WW:LET MV1=J:LET MV2=J:L
ET SW1=SW:LET KK=K
1880 FOR I=1 TO 24:LET B(I)=B2(I):NEXT I
1890 NEXT J
1900 IF FL=1 THEN LET FL=0:GOTO 2410
1910 IF FL=2 THEN LET FL=0:GOTO 1960
1920 IF FL=3 THEN LET FL=0:GOTO 1970
1930 RETURN
1940 REM ** SOME ILLEGAL MOVE(S) **
1950 LET FL=2:LET K=1:GOTO 1660
1960 LET FL=3:LET K=2:GOTO 1660
1970 IF MX=-20 THEN GOSUB 3580:GOTO 2200
1980 IF KK=2 THEN GOSUB 3520:LET Z$="Y":GOSUB 3610
:LET Z$=""
1990 IF B(C(MV1)-M(1))=1 THEN LET B(C(MV1)-M(1))=0
:LET PB=PB+1
2000 LET B(C(MV1))=B(C(MV1))+1:LET B(C(MV1)-M(1))=
B(C(MV1)-M(1))-1
2010 GOSUB 3580
2020 IF MVS=1 THEN LOCATE #3,7,6:PRINT #3,"DIE 3 =
 ";CHR$(64+C(MV1)):GOTO 2040
2030 LOCATE #3,7,4:PRINT #3,"DIE 1 = ";CHR$(64+C(M
V1))
2040 GOSUB 3960
2050 LET C(MV1)=C(MV1)-M(1)
2060 LET I=2
2070 IF MVS=1 THEN LET MVS=0:LET I=4
2080 LET FL1=1:GOTO 2740
2090 RETURN
2100 REM ** COMPUTER OFF BAR **
2110 CLS #3:PRINT #3:PRINT #3," COMPUTER ON THE BA
R"
2120 FOR Z=1 TO 1000:NEXT Z
2130 GOSUB 3550
2140 IF B(25-M(1))<0 THEN GOTO 2230
2150 IF B(25-M(2))<0 THEN GOSUB 3520:LET Z$="Y":GO
SUB 3610:LET Z$="":GOTO 2230
2160 IF B(25-M(1))=1 THEN GOTO 2230
```

```
2170 IF B(25-M(2))=1 THEN GOSUB 3520:LET Z$="Y":GO
SUB 3610:LET Z$="":GOTO 2230
2180 IF B(25-M(1))=0 THEN GOTO 2230
2190 IF B(25-M(2))=0 THEN GOSUB 3520:LET Z$="Y":GO
SUB 3610:LET Z$="":GOTO 2230
2200 CLS #3:LOCATE #3,5,4:PRINT #3,"NO LEGAL MOVE"
2210 PRINT #3,"   COMPUTER PASSES"
2220 FOR Z=1 TO 1000:NEXT Z:RETURN
2230 REM ** MOVE OFF BAR **
2240 IF B(25-M(1))=1 THEN LET B(25-M(1))=0:LET PB=
PB+1
2250 IF B(25-M(1))<>1 THEN GOTO 2290
2260 FOR I=1 TO 15
2270 IF P(I)=25-M(1) THEN LET P(I)=25
2280 NEXT I
2290 LET B(25-M(1))=B(25-M(1))-1
2300 LET CB=CB-1
2310 FOR I=1 TO 15
2320 IF C(I)=25 THEN LET C(I)=25-M(1):GOTO 2340
2330 NEXT I
2340 GOSUB 3960
2350 FOR Z=1 TO 1000:NEXT Z
2360 CLS #3:PRINT #3:PRINT #3,"   COMPUTER'S TURN"
2370 IF CB>0 THEN GOTO 2510
2380 GOSUB 3550
2390 FOR II=1 TO 24:LET B2(II)=B(II):NEXT II
2400 LET MV1=0:LET K=2:LET FL=1:GOTO 1660
2410 IF MV1=0 THEN GOTO 2200
2420 IF B(C(MV2)-M(2))=1 THEN LET B(C(MV2)-M(2))=0
:LET PB=PB+1
2430 LET B(C(MV2))=B(C(MV2))+1:LET B(C(MV2)-M(2))=
B(C(MV2)-M(2))-1
2440 GOSUB 3580
2450 IF MVS=1 THEN LOCATE #3,7,7:PRINT #3,"DIE 4 =
 ";CHR$(64+C(MV2)):LET MVS=0:GOTO 2470
2460 LOCATE #3,7,5:PRINT #3,"DIE 2 = ";CHR$(64+C(M
V2))
2470 GOSUB 3960
2480 LET C(MV2)=C(MV2)-M(2)
2490 IF MOVES=4 THEN LET MVS=1:LET MOVES=2:GOTO 11
80
2500 RETURN
2510 CLS #3:PRINT #3:PRINT #3," COMPUTER ON THE BA
R"
2520 GOSUB 3550
2530 IF B(25-M(2))<0 THEN GOTO 2570
2540 IF B(25-M(2)=1 THEN GOTO 2570
2550 IF B(25-M(2)=0 THEN GOTO 2570
2560 GOTO 2200
2570 IF B(25-M(2))=1 THEN LET B(25-M(2))=0:LET PB=
PB+1
```

```
2580 IF B(25-M(2))<>1 THEN GOTO 2620
2590 FOR I=1 TO 15
2600 IF P(I)=25-M(2) THEN LET P(I)=25
2610 NEXT I
2620 LET B(25-M(2))=B(25-M(2))-1
2630 LET CB=CB-1
2640 FOR I=1 TO 15
2650 IF C(I)=25 THEN LET C(I)=25-M(2):GOTO 2670
2660 NEXT I
2670 GOSUB 3960
2680 GOTO 2490
2690 REM ** COMPUTER OFF BOARD **
2700 PRINT #3
2710 GOSUB 3580
2720 IF MVS=1 THEN FOR I=3 TO 4
2730 IF MVS=0 THEN FOR I=1 TO MOVES
2740 IF B(M(I))<0 THEN LET B(M(I))=B(M(I))+1:LET C
OB=COB+1:LET J=M(I):GOTO 2890
2750 FOR K=M(I)+1 TO 8
2760 IF B(K)<0 THEN GOTO 2820
2770 NEXT K
2780 FOR J=M(I)-1 TO 1 STEP -1
2790 IF B(J)>-1 THEN GOTO 2810
2800 LET B(J)=B(J)+1:LET COB=COB+1:GOTO 2890
2810 NEXT J
2820 FOR J=8 TO M(I)+1 STEP -1
2830 IF B(J)>-1 THEN GOTO 2870
2840 IF B(J-M(I))>1 THEN GOTO 2870
2850 IF B(J-M(I))=1 THEN LET B(J-M(I))=-1:LET B(J)
=B(J)+1:LET PB=PB+1:GOTO 2890
2860 LET B(J-M(I))=B(J-M(I))-1:LET B(J)=B(J)+1:GOT
O 2890
2870 NEXT J
2880 GOTO 2980
2890 LOCATE #3,7,I+3:PRINT #3,"DIE";I;"= ";CHR$(64
+J):FOR Z=1 TO 1000:NEXT Z:GOSUB 3960
2900 FOR II=1 TO 15
2910 IF C(II)=J AND J-M(I)<1 THEN LET C(II)=-1:GOT
O 2940
2920 IF C(II)=J THEN LET C(II)=J-M(I)
2930 NEXT II
2940 IF FL1=1 THEN LET FL1=0:RETURN
2950 IF COB=15 THEN RETURN
2960 NEXT I
2970 RETURN
2980 LOCATE #3,7,I+3:PRINT #3,"DIE";I;"= PASS"
2990 IF FL1=1 THEN LET FL1=0:RETURN
3000 NEXT I
3010 FOR Z=1 TO 1000:NEXT Z
3020 RETURN
3030 REM ** COMPUTER DOUBLE **
```

```
3040 LET YD=0
3050 IF COB-POB>4 THEN LET YD=1
3060 IF PB-CB>=3 THEN LET YD=1
3070 FOR II=1 TO 15
3080 IF C(II)>6 THEN GOTO 3110
3090 NEXT II
3100 IF YD=0 THEN RETURN
3110 LET TP=0:LET TC=0
3120 FOR I=1 TO 24
3130 IF B(I)<0 THEN LET TC=TC+ABS(B(I)*I)
3140 IF B(I)>0 THEN LET TP=TP+ABS(B(I)*I)
3150 NEXT I
3160 LET TC=TC-15
3170 LET TP=360-TP
3180 IF TP-TC>25 AND POB=0 THEN LET YD=1
3190 IF YD=0 THEN RETURN
3200 CLS #3:PRINT #3:PRINT #3,"   COMPUTER'S TURN"
3210 PRINT #3:PRINT #3,"   COMPUTER OFFERS":PRINT
#3,"  THE  DOUBLING DIE"
3220 PRINT #3:PRINT #3," DO YOU ACCEPT (Y/N)?"
3230 LET QQ$=INKEY$:IF QQ$="" THEN GOTO 3230
3240 IF QQ$="Y" THEN LET PTS=PTS*2:LET STATUS=STAT
US+1:GOSUB 3880:RETURN
3250 IF QQ$="N" THEN LET Z$="COMPUTER":GOTO 3270
3260 GOTO 3230
3270 REM ** GAME OVER **
3280 CLS #3
3290 PRINT #3:PRINT #3:PRINT #3,"      GAME OVER"
3300 IF Z$="PLAYER" THEN PRINT #3:PRINT #3,"     P
LAYER WINS":GOTO 3320
3310 PRINT #3:PRINT #3,"    ";Z$;" WINS"
3320 IF PTS<>1 THEN PRINT #3,"     BY";PTS;"POINTS
"
3330 IF PTS=1 THEN PRINT #3,"      BY 1 POINT"
3340 LOCATE 1,1:END
3350 REM ** DRAW BAR **
3360 CLS #2
3370 LET Y=182
3380 IF PB=0 THEN GOTO 3440
3390 PLOT 650,650,2
3400 FOR II=1 TO PB
3410 MOVE 240,Y:TAG:PRINT CHR$(240);:TAGOFF
3420 LET Y=Y-8
3430 NEXT II
3440 IF CB=0 THEN GOTO 3510
3450 PLOT 650,650,1
3460 LET Y=80
3470 FOR II=1 TO CB
3480 MOVE 240,Y:TAG:PRINT CHR$(241);:TAGOFF
3490 LET Y=Y+8
3500 NEXT II
```

```
3510 RETURN
3520 REM ** SWITCH DICE **
3530 LET MM=M(1):LET M(1)=M(2):LET M(2)=MM
3540 RETURN
3550 REM ** THINK MESSAGE **
3560 LOCATE #3,3,7:PRINT #3,"AMSTRAD  THINKING"
3570 RETURN
3580 REM ** REMOVE MESSAGE **
3590 LOCATE #3,3,7:PRINT #3,STRING$(17," ")
3600 RETURN
3610 REM ** CLEAR DICE **
3620 PAPER #4,0
3630 FOR II=2 TO 4
3640 LOCATE #4,2,II:PRINT #4,"   ";
3650 LOCATE #4,6,II:PRINT #4,"   ";
3660 NEXT II
3670 IF Z$="Y" THEN LET Z$="":GOSUB 3780:RETURN
3680 GOSUB 3700
3690 RETURN
3700 REM ** ROLL DICE **
3710 FOR I=1 TO 4:LET M(I)=0:NEXT I
3720 LET M(1)=1+INT(6*(RND(1)))
3730 LET M(2)=1+INT(6*(RND(1)))
3740 LET MOVES=2
3750 IF M(1)=M(2) THEN LET MOVES=4:LET M(3)=M(1):L
ET M(4)=M(1)
3760 GOSUB 3780
3770 RETURN
3780 REM ** DRAW SPOTS **
3790 LET Y=2
3800 FOR I=1 TO 2
3810 LET X=2+4*(I-1)
3820 IF M(I)=1 OR M(I)=3 OR M(I)=5 THEN LOCATE #4,
X+1,Y+1:PRINT #4,CHR$(144);
3830 IF M(I)=4 OR M(I)=5 OR M(I)=6 THEN LOCATE #4,
X,Y:PRINT #4,CHR$(144);" ";CHR$(144):LOCATE #4,X,Y
+2:PRINT #4,CHR$(144);" ";CHR$(144)
3840 IF M(I)=6 THEN LOCATE #4,X,Y+1:PRINT #4,CHR$(
144);" ";CHR$(144)
3850 IF M(I)=2 OR M(I)=3 THEN LOCATE #4,X,Y:PRINT
#4,CHR$(144):LOCATE #4,X+2,Y+2:PRINT #4,CHR$(144)
3860 NEXT I
3870 RETURN
3880 REM ** DRAW DOUBLING DIE **
3890 PAPER #5,0:PEN #5,1
3900 FOR II=2 TO 4
3910 LOCATE #5,2,II:PRINT #5,"   "
3920 NEXT II
3930 IF PTS<10 THEN LOCATE #5,2,3:PRINT #5,PTS;:RE
TURN
3940 PLOT 650,650,1:TAG:MOVE 537,126:PRINT USING "
```

```
##";PTS;:TAGOFF
3950 RETURN
3960 REM ** POSITION PIECES **
3970 GOSUB 3350
3980 LET X=48
3990 FOR II=1 TO 6
4000 LET Y=212
4010 IF B(II)=0 THEN MOVE X,Y:TAG:PRINT " ";:TAGOF
F:GOTO 4090
4020 PLOT 650,650,1
4030 IF B(II)>0 THEN PLOT 650,650,2
4040 FOR JJ=1 TO ABS(B(II))
4050 IF JJ>6 THEN GOTO 4090
4060 MOVE X,Y:TAG:PRINT CHR$(240);:TAGOFF
4070 LET Y=Y-8
4080 NEXT JJ
4090 LET X=X+32
4100 NEXT II
4110 LET X=272
4120 FOR II=7 TO 12
4130 LET Y=212
4140 IF B(II)=0 THEN MOVE X,Y:TAG:PRINT " ";:TAGOF
F:GOTO 4220
4150 PLOT 650,650,1
4160 IF B(II)>0 THEN PLOT 650,650,2
4170 FOR JJ=1 TO ABS(B(II))
4180 IF JJ>6 THEN GOTO 4220
4190 MOVE X,Y:TAG:PRINT CHR$(240);:TAGOFF
4200 LET Y=Y-8
4210 NEXT JJ
4220 LET X=X+32
4230 NEXT II
4240 LET X=432
4250 FOR II=13 TO 18
4260 LET Y=58
4270 IF B(II)=0 THEN MOVE X,Y:TAG:PRINT " ";:TAGOF
F:GOTO 4350
4280 PLOT 650,650,1
4290 IF B(II)>0 THEN PLOT 650,650,2
4300 FOR JJ=1 TO ABS(B(II))
4310 IF JJ>6 THEN GOTO 4350
4320 MOVE X,Y:TAG:PRINT CHR$(241);:TAGOFF
4330 LET Y=Y+8
4340 NEXT JJ
4350 LET X=X-32
4360 NEXT II
4370 LET X=208
4380 FOR II=19 TO 24
4390 LET Y=58
4400 IF B(II)=0 THEN MOVE X,Y:TAG:PRINT " ";:TAGOF
F:GOTO 4480
```

```
4410 PLOT 650,650,1
4420 IF B(II)>0 THEN PLOT 650,650,2
4430 FOR JJ=1 TO ABS(B(II))
4440 IF JJ>6 THEN GOTO 4480
4450 MOVE X,Y:TAG:PRINT CHR$(241);:TAGOFF
4460 LET Y=Y+8
4470 NEXT JJ
4480 LET X=X-32
4490 NEXT II
4500 RETURN
4510 REM ** LEGALITY ROUTINE **
4520 LET LEGAL=1
4530 IF G1/2<>INT(G1/2) AND A-M(I)<1 THEN LET LEGA
L=0:RETURN
4540 IF G1/2=INT(G1/2) AND A+M(I)>24 AND TST<>15 T
HEN LET LEGAL=0:RETURN
4550 IF B(A)<=0 AND INT(G1/2)=G1/2 THEN LET LEGAL=
0:RETURN
4560 IF G1/2<>INT(G1/2) OR TST <>15 OR M(I)+A<25 T
HEN  GOTO 4620
4570 IF TST=15 AND 25-M(I)=A THEN RETURN
4580 FOR II=18 TO A-1
4590 IF B(II)>0 THEN LET LEGAL=0:RETURN
4600 NEXT II
4610 RETURN
4620 IF G1/2=INT(G1/2) AND TST=15 AND A+M(I)>24 TH
EN RETURN
4630 IF B(A)>=0 AND INT(G1/2)<>G1/2 THEN LET LEGAL
=0:RETURN
4640 IF G1/2=INT(G1/2) THEN IF B(A+M(I))<-1 THEN L
ET LEGAL=0:RETURN
4650 IF G1/2<>INT(G1/2) THEN IF B(A-M(I))>1 THEN L
ET LEGAL=0:RETURN
4660 IF G1/2=INT(G1/2) AND PB>0 THEN LET LEGAL=0:R
ETURN
4670 IF G1/2<>INT(G1/2) AND CB>0 THEN LET LEGAL=0:
RETURN
4680 RETURN
4690 REM ** WEIGHTING **
4700 LET W=0
4710 IF TST>11 AND C(J)>7 THEN LET W=W+5
4720 IF TST>11 AND C(K)>7 THEN LET W=W+5
4730 IF C(J)=C(K) AND B(C(J))<7 THEN LET W=W-1
4740 IF B(C(K)-M(1))=1 THEN LET W=W+4
4750 IF C(J)-M(2)<>C(K)-M(1) AND B(C(J)-M(2))=1 TH
EN LET W=W+4
4760 FOR II=1 TO 24
4770 LET B1(II)=B(II)
4780 NEXT II
4790 IF B1(C(K)-M(1))=1 THEN LET B1(C(K)-M(1))=0
4800 LET B1(C(K))=B1(C(K))+1:LET B1(C(K)-M(1))=B1(
```

```
C(K)-M(1))-1
4810 IF B1(C(J)-M(2))=1 THEN LET B1(C(J)-M(2))=0
4820 LET B1(C(J))=B1(C(J))+1:LET B1(C(J)-M(2))=B1(
C(J)-M(2))-1
4830 FOR II=1 TO 24
4840 IF B1(II)=-1 THEN LET W=W-3
4850 IF B1(II)<0 AND II>18 THEN LET W=W-((II-18)*A
BS(B1(II)))
4860 IF II<=6 AND B(II)<0 THEN LET W=W+1
4870 NEXT II
4880 RETURN
4890 REM ** INITIALISE GAME **
4900 LET CH=5:LET CB=0:LET PH=5:LET PB=0
4910 FOR I=1 TO 15
4920 READ C(I)
4930 NEXT I
4940 FOR I=1 TO 15
4950 READ P(I)
4960 NEXT I
4970 FOR I=1 TO 24
4980 READ B(I)
4990 NEXT I
5000 RETURN
5010 DATA 24,24,13,13,13,13,13,9,9,9,6,6,6,6,6,1,1
,12,12,12,12,12,16,16,16,19,19,19,19,19
5020 DATA 2,0,0,0,0,-5,0,0,-3,0,0,5,-5,0,0,3,0,0,5
,0,0,0,0,-2
5030 REM ** DRAW BOARD **
5040 PLOT 650,650,1
5050 CLS #1
5060 PLOT 30,30:DRAW 30,224:DRAW 464,224:DRAW 464,
30:DRAW 30,30
5070 PLOT 270,384:DRAW 608,384:DRAW 608,254:DRAW 2
70,254:DRAW 270,384
5080 PLOT 62,352:DRAW 208,352:DRAW 208,270:DRAW 62
,270:DRAW 62,352
5090 PLOT 510,160:DRAW 592,160:DRAW 592,78:DRAW 51
0,78:DRAW 510,160
5100 FOR Z=1 TO 6
5110 FOR Y=12 TO 16
5120 LOCATE 2+2*Z,Y:PRINT " ";
5130 LOCATE 16+2*Z,Y:PRINT " ";
5140 NEXT Y
5150 FOR Y=19 TO 23
5160 LOCATE 16-2*Z,Y:PRINT " ";
5170 LOCATE 30-2*Z,Y:PRINT " ";
5180 NEXT Y
5190 NEXT Z
5200 FOR Z=46 TO 430 STEP 32
5210 IF Z=238 THEN GOTO 5240
5220 PLOT Z,224:DRAWR 0,-82:DRAWR 18,0:DRAWR 0,82
```

```
5230 PLOT Z,30:DRAWR 0,82:DRAWR 18,0:DRAWR 0,-82
5240 NEXT Z
5250 FOR Z=48 TO 208 STEP 32:MOVE Z,244:TAG:PRINT
CHR$(65+(Z-48)/32);:TAGOFF:NEXT Z
5260 FOR Z=272 TO 432 STEP 32:MOVE Z,244:TAG:PRINT
 CHR$(65+(Z-80)/32);:TAGOFF:NEXT Z
5270 FOR Z=432 TO 272 STEP -32:MOVE Z,22:TAG:PRINT
 CHR$(77+(432-Z)/32);:TAGOFF:NEXT Z
5280 FOR Z=208 TO 48 STEP -32:MOVE Z,22:TAG:PRINT
CHR$(76+(432-Z)/32);:TAGOFF:NEXT Z
5290 CLS #2
5300 PLOT 238,46:DRAW 238,208:DRAW 256,208:DRAW 25
6,46:DRAW 238,46
5310 PLOT 78,336:DRAW 78,286:DRAW 128,286:DRAW 128
,336:DRAW 78,336
5320 PLOT 526,144:DRAW 526,94:DRAW 576,94:DRAW 576
,144:DRAW 526,144
5330 PLOT 142,336:DRAW 142,286:DRAW 192,286:DRAW 1
92,336:DRAW 142,336
5340 RETURN
5350 REM ** USER-DEFINED GRAPHICS **
5360 SYMBOL AFTER 240
5370 SYMBOL 240,0,&18,&18,0,0,0,0,0
5380 SYMBOL 241,0,0,0,0,0,&18,&18,0
5390 RETURN
```

11. Draughts

This final program is devoted to the game of Draughts. You will be given the opportunity to challenge the computer and demonstrate your superiority in one of the most complicated of board games.

As a computer owner, you have no doubt encountered versions of this game before, but it is very likely that they were much shorter and did not play to such a high standard. This program has been designed to play the best possible game while keeping the time taken for each move down to well below one minute.

Playing instructions

When the game is executed you will be offered the first move, after which the game progresses in the normal manner with each player taking alternate turns, you moving the 'red' pieces and the computer the 'blue'.

The program obeys all the normal rules of the game, including 'huffing' which means that if it is possible to take an opponent's piece then this must be done, otherwise the opponent may remove your offending piece (huff the piece) from the board.

The game continues, with the time taken by both players being indicated at the bottom of the screen, until one player loses all his pieces or there is no legal move available, at which time the game is declared to be stalemate. Throughout the game full legality checks will be performed on all your moves and it will therefore be necessary to use skill as opposed to unfair tactics in order to defeat the computer.

The program

As outlined in Chapter 1, Draughts (like Chess) is a difficult game to implement well on a computer, especially if we are restricted to using Basic as opposed to machine code. The major problem is the number of possible legal moves to be considered at any stage

of the game. If we were to employ a four-ply move tree in full then the number of combinations to consider would be astronomical, with the computer taking hours if not days to make a single move.

In order to keep the time taken for each move down to a realistic level – below one minute was considered satisfactory for this program – the look-ahead depth had to be severely restricted or the move tree heavily pruned. The second alternative was considered the most suitable as under certain conditions it is necessary to consider a move in more detail.

The selective pruning of the move tree can be observed very clearly as the game is played, since the time taken for each move by the computer is far from constant, varying according to the depth of the search. A good example of this can be seen during the end game, where although one would expect the computer to speed up due to the limited number of pieces, this is not the case in practice as less pruning is done to the tree.

Close inspection of the flowchart and comments associated with this program indicate that a minimax type of heuristic is employed, and the weight W for each move is altered in several of the subroutines. With care it should be a relatively simple matter to change some of the weightings or add new ones in an attempt to make the computer play a better game.

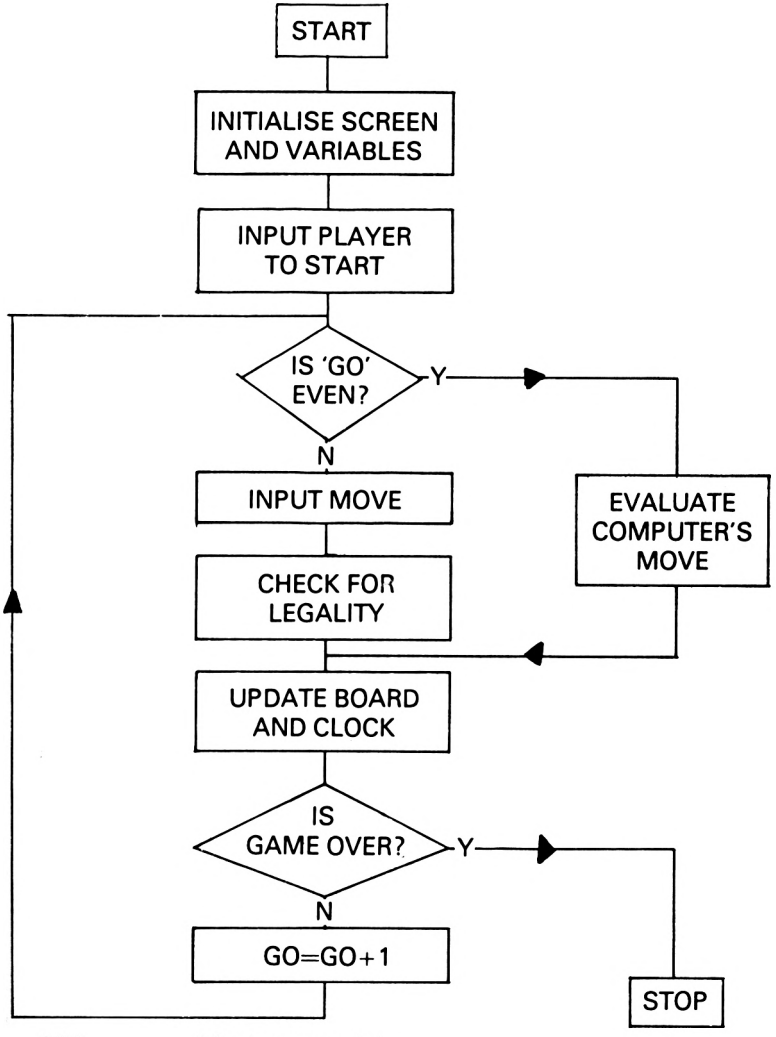
Flowchart

The flowchart (opposite) represents the general operation of the program.

Variables

The following table represents the major variables used in the program and should help you to understand how the program operates and make changes to the evaluation routines.

B$(x,y)	Array storing current position of game
COPY$(x,y)	Copy of array B$
DIR$	Direction of scan (up or down)
GO	Move number
Z$	Winner declaration
CC	Number of computer's pieces
PC	Number of player's pieces
CK	Number of computer's kings
PK	Number of player's kings
W	Weight, calculated for each move



MX	Maximum weight
LEGAL	Variable set to 1 if a move is legal
JUMP	Multiple jump variable
M	Variable used to store the type of piece being considered
REP	Variable used for finding repetition
REPEAT	Flag set to 1 when the same move is made four times by the computer
CBD	Variable used in the end game to find distance from player's piece

Comments on the program

The program relies very heavily on the use of subroutines and this should make it easy to follow and adapt. Each routine is clearly labelled in the listing and a breakdown of each is given below.

Lines 10-370
This routine initialises variables, dimensions arrays and designs the screen format.

Lines 380-420
A welcome message is displayed on the screen, with the player inputting who is to have the first move.

Lines 430-500
This represents the whole of the main program. The value of the variable GO is checked and the appropriate routines are called.

Lines 510-1600
This very long subroutine is where the computer selects and then makes the best possible move. It operates by considering every move down to a depth of one. The move tree is then pruned, and other routines looking to a much greater depth are executed when appropriate.

The group of lines 1010-1300 are of great importance if you are considering changing the program in order to play a new strategy. It is in this section that various patterns are considered and the weight W is adjusted depending on whether the current position is good or bad. Throughout these lines, use is made of the variable M, the value of which indicates the piece being considered, and the array B$(x,y) which contains a code representing the piece at every position on the board.

Codes for M and B$(x,y)
1 player's piece
-1 computer's piece
2 player's king
-2 computer's king

As an example of how this section works, consider line 1050:

```
1050 IF M=-1 AND Y1=8 AND CC>7 THEN LET W=W-1
```

This decreases the weighting by 1 of any move off the back row if the number of computer's pieces is greater than seven.

Lines 1610-1990
This routine is called from the main program when the variable GO is an odd number. It allows the player to make a move, which is checked for legality and then displayed on the board.

Lines 2000-2170
A routine used to see if a move made by the player or considered by the computer is legal.

Lines 2180-2280
This short routine is used to investigate if multiple moves are possible.

Lines 2290-2380
This executes the multiple jumps discovered in the previous routine.

Lines 2390-2520
A routine used to consider multiple jumps in the look-ahead section where the computer will recapture after one of its pieces has been removed.

Lines 2530-2860
This is the main subroutine for considering the move tree below the first level. It is not called for every possible move, only those which the computer considers to be appropriate.

Lines 2870-2990
A short routine to consider if the computer is in an attacking position at the end of a series of moves.

Lines 3000-3140
Another routine to consider if the player is in an attacking position after a series of moves.

Lines 3150-3190
The interrupt routine used to update the computer's clock.

Lines 3200-3240
The interrupt routine used to update the player's clock.

Lines 3250-3540
This is used to display the current position of the board on the screen.

Lines 3550-3770
Routine to initialise the board at the beginning of the game.

Lines 3780-3790
A routine to set the stalemate flag if there are no legal moves.

Lines 3800-3990
One of the two routines used to control the huffing procedure.

Lines 4000-4150
The second of the two huffing routines.

Lines 4160-4270
As with any board game, the end-game is of paramount importance. This program uses a special routine to evaluate the best possible move at this stage.

Lines 4280-4380
A routine used to check if the game has reached the end-game position, when a new set of criteria is used to evaluate the best possible move.

Lines 4390-4530
This routine is used at the end-game stage to check on the repetition of moves.

Lines 4540-4660
The final routine used when the game is over to declare the winner and offer another game.

The listing

The program should be entered very carefully and then saved on to cassette by following the procedure outlined in the Introduction.

The game should be entered and played in the caps lock mode. All spaces within the text should be entered as they appear.

```
10 LET D=REMAIN(1)
20 LET PC=12:LET CC=12:LET PK=0:LET CK=0
30 LET DIR$="U"
40 DIM B$(9,9),COPY$(9,9),D$(9,9),HS$(9,9),HP$(9,9
)
50 INK 0,1
```

```
60 INK 1,6
70 PAPER 0
80 LET MOV=1
90 SYMBOL AFTER 247
100 SYMBOL 251,0,0,0,3,7,15,15,31
110 SYMBOL 253,31,15,15,7,3,0,0,0
120 SYMBOL 252,0,0,0,192,224,240,240,248
130 SYMBOL 254,248,240,240,224,192
140 SYMBOL 247,0,0,0,3,7,15,14,28
150 SYMBOL 249,28,14,15,7,3,0,0,0
160 SYMBOL 248,0,0,0,192,224,240,112,56
170 SYMBOL 250,56,112,240,224,192
180 CLS
190 LET REP=0:LET REPEAT=0
200 GOSUB 3550
210 GOSUB 3250
220 WINDOW #2,20,38,5,12
230 PAPER#2,2:CLS#2
240 WINDOW#3,4,12,20,23
250 PAPER#3,2:PEN#3,0:CLS#3
260 WINDOW#4,25,33,20,23
270 PAPER#4,2:PEN#4,1:CLS#4
280 LOCATE #3,1,1
290 PRINT#3,"COMP TIME"
300 LOCATE #4,1,1
310 PRINT#4,"PLAY TIME"
320 PEN#3,3
330 LOCATE #3,3,3
340 PRINT#3," 0:00"
350 PEN#4,3
360 LOCATE #4,3,3
370 PRINT#4," 0:00"
380 LOCATE#2,1,2
390 PRINT#2,"WELCOME TO DRAUGHTS"
400 PRINT#2,"DO YOU WANT TO"
410 INPUT#2,"MOVE FIRST ? ",Q$
420 IF LEFT$(Q$,1)="Y" THEN LET GO=1 ELSE LET GO=0
430 REM **PLAY ROUTINE STARTS HERE**
440 IF GO/2=INT(GO/2) THEN GOSUB 510 ELSE GOSUB 16
10
450 GOSUB 2510
460 IF GO/2<>INT(GO/2) THEN GOSUB 4390
470 LET GO=GO+1
480 IF CC=0 THEN LET Z$="PLAYER":GOTO 4540
490 IF PC=0 THEN LET Z$="COMPUTER":GOTO 4540
500 GOTO 440
510 REM **COMPUTER'S MOVE**
520 IF CK>=CC/2 THEN LET EG=1 ELSE LET EG=0
530 IF EG>0 THEN GOSUB 4280
540 IF EG/2<>INT(EG/2) THEN LET DIR$="D" ELSE LET
DIR$="U"
```

```
550 IF EG=0 AND CC<=6 THEN LET DIR$="D"
560 LET M=0
570 IF DIR$="U" THEN LET WK=-2 ELSE LET WK=2
580 LET X1=0:LET X2=0:LET Y1=0:LET Y2=0
590 LET AGAIN=0
600 LET MX=-100
610 EVERY 50,1 GOSUB 3150
620 FOR I=1 TO 8
630 FOR J=1 TO 8
640 LET COPY$(J,I)=B$(J,I)
650 NEXT J,I
660 CLS#2:LOCATE#2,2,2
670 PRINT#2,"COMPUTER'S MOVE"
680 PRINT#2,"FROM:"
690 PRINT#2,"  TO:"
700 IF MOV=1 THEN LET X1=2:LET Y1=6:LET X2=1:LET Y
2=5:LET M=-1:GOTO 1480
710 FOR G=2 TO 1 STEP-1
720 IF DIR$="U" THEN LET S=8:F=1:ST=-1
730 IF DIR$="D" THEN LET S=1:F=8:ST=1
740 FOR Y1=S TO F STEP ST
750 IF Y1/2=INT(Y1/2) THEN LET S=1:LET F=8:LET ST=
1
760 IF Y1/2<>INT(Y1/2) THEN LET S=8:LET F=1:LET ST
=-1
770 FOR X1=S TO F STEP ST
780 IF B$(X1,Y1)="-1" THEN LET M=-1:GOTO 810
790 IF B$(X1,Y1)="-2" THEN LET M=-2:GOTO 810
800 LET M=0:GOTO 890
810 IF Y1/2 = INT (Y1/2) THEN LET Q=G ELSE LET Q=-
G
820 FOR X2=X1-G TO X1+G STEP 2*G
830 IF X2>8 OR X2<1 THEN GOTO 880
840 IF M=-1 THEN LET Y2=Y1-G ELSE FOR Y2=Y1-G TO Y
1+G STEP 2*G
850 GOSUB 2000
860 IF LEGAL=1 THEN GOTO 930
870 IF M=-2 THEN NEXT Y2
880 NEXT X2
890 NEXT X1
900 NEXT Y1
910 IF HUFF=0 THEN NEXT G
920 GOTO 1420
930 LET W=0
940 LET F$=CHR$(X1+64)+CHR$(Y1+48)
950 IF X2=0 THEN LET G$="":GOTO 970
960 LET G$=CHR$(X2+64)+CHR$(Y2+48)
970 LET MM=VAL(B$(X1,Y1)):LET X11=X1:LET X22=X2:LE
T Y11=Y1:LET Y22=Y2:LET GG=G
980 LET M=VAL(B$(X1,Y1)):LET G=GG:LET B$(X1,Y1)="0
"
```

```
990 IF ABS(X1-X2)=1 THEN LET W=W-1
1000 IF M=-1 THEN LET B$(X2,Y2)="-1" ELSE LET B$(X
2,Y2)="-2"
1010 IF B$(X2-1,Y2-1)="1" AND B$(X2+1,Y2+1)="1" AN
D Y2=2 THEN LET W=W+2
1020 IF B$(X2+1,Y2-1)="1" AND B$(X2-1,Y2+1)="1" AN
D Y2=2 THEN LET W=W+2
1030 IF M=-1 AND B$(X2-1,Y2-1)="1" THEN LET W=W+1
1040 IF M=-1 AND B$(X2+1,Y2-1)="1" THEN LET W=W+1
1050 IF M=-1 AND Y1=8 AND CC>7 THEN LET W=W-1
1060 IF M=-1 AND Y1>1 THEN IF  VAL(B$(X1,Y1-2))>0
THEN LET W=W-2
1070 IF M=-2 AND Y2>Y1 AND EG>0 THEN LET W=W+WK
1080 IF M=-1 AND EG>0 THEN LET W=W+2
1090 IF ABS(X2-X1)=2 AND B$((X1+X2)/2,(Y1+Y2)/2)="
2" THEN LET W=W+3
1100 IF M=-2 AND Y1=1 THEN LET W=W+1
1110 IF X2=1 AND Y2=3 AND M=-1 THEN LET W=W+2
1120 IF M=-2 AND X2=6 AND B$(8,Y2)="2" THEN LET W=
W+3
1130 IF M=-2 AND X1=6 AND B$(8,Y2)="2" THEN LET W=
W-3
1140 IF M=-2 AND Y2=1 THEN LET W=W-1
1150 IF M=-2 AND X2=3 AND B$(1,Y2)="2" THEN LET W=
W+3
1160 IF M=-2 AND X1=3 AND B$(1,Y2)="2" THEN LET W=
W-3
1170 IF M=-2 AND Y2=6 AND B$(X2,8)="2" THEN LET W=
W+3
1180 IF M=-2 AND Y1=6 AND B$(X2,8)="2" THEN LET W=
W-3
1190 IF M=-2 AND Y2=3 AND B$(X2,1)="2" THEN LET W=
W+3
1200 IF M=-2 AND Y1=3 AND B$(X2,1)="2" THEN LET W=
W-3
1210 IF REPEAT=1 AND X1=RX1 AND X2=RX2 AND Y1=RY1
AND Y2=RY2 THEN LET W=W-2
1220 LET MR=M:LET X1M=X1:LET Y1M=Y1:LET X2M=X2:LET
 Y2M=Y2
1230 IF M=-2 AND ABS(X2-X1)=1 THEN LET W=W-1
1240 IF M=-1 AND Y1=8 AND B$(X1,6)="1" THEN LET W=
W-3
1250 IF Y2<>6 OR X2<=2 THEN GOTO 1270
1260 IF Y2=6 AND B$(X2-2,Y2)="1" AND B$(X2-2,8)="-
1" THEN LET W=W-3
1270 IF Y2<>6 OR X2>=7 THEN GOTO 1290
1280 IF Y2=6 AND B$(X2+2,Y2)="1" AND B$(X2+2,8)="-
1" THEN LET W=W-3
1290 IF ABS(X2-X1)=2 THEN LET HUFF=1:LET B$((X1+X2
)/2,(Y1+Y2)/2)="0":LET W=W+5:LET JUMPS=JUMPS+1:GOS
UB 2180
```

```
1300 IF AGAIN=1 THEN GOTO 980
1310 IF JUMPS=0 THEN GOSUB 2870
1320 IF MR=-2 THEN GOSUB 4160
1330 IF W>MX OR EG>0 THEN FOR I=1 TO 8:FOR J=1 TO
8:LET D$(J,I)=B$(J,I):NEXT J,I: GOSUB 2530
1340 LET M=MM:LET X1=X11:LET Y1=Y11:LET X2=X22:LET
 Y2=Y22:LET G=GG:IF W<=MX THEN GOTO 1370
1350 LET A1=X1:A2=X2:B1=Y1:B2=Y2:JUMP=JUMPS
1360 LET MX=W
1370 FOR HJ=1 TO 8
1380 FOR HK=1 TO 8
1390 LET B$(HK,HJ)=COPY$(HK,HJ)
1400 NEXT HK,HJ
1410 LET W=0:LET JUMPS=0:GOTO 870
1420 LET HUFF=0:FOR HJ=1 TO 8
1430 FOR HK=1 TO 8
1440 LET B$(HK,HJ)=COPY$(HK,HJ)
1450 NEXT HK,HJ
1460 IF MX=-100 THEN GOTO 3780
1470 LET X1=A1:LET X2=A2:LET Y1=B1:LET Y2=B2:LET J
UMPS=JUMP:LET M=VAL(B$(X1,Y1))
1480 LET F$=CHR$(X1+64)+CHR$(Y1+48)
1490 LET G$=CHR$(X2+64)+CHR$(Y2+48)
1500 LOCATE#2,6,3:PRINT#2,F$
1510 LOCATE#2,6,4:PRINT#2,G$
1520 LET RX1=X1:LET RX2=X2:LET RY1=Y1:LET RY2=Y2
1530 LET B$(X1,Y1)="0"
1540 IF M=-1 THEN LET B$(X2,Y2)="-1" ELSE LET B$(X
2,Y2)="-2"
1550 IF ABS(X2-X1)=2 THEN LET B$((X1+X2)/2,(Y1+Y2)
/2)="0":LET JUMPS=JUMPS-1:LET PC=PC-1
1560 GOSUB 3250
1570 IF JUMPS>0 THEN GOSUB 2290:GOTO 1480
1580 IF Y2=1 AND B$(X2,Y2)="-1" THEN LET B$(X2,Y2)
="-2":LET CK=CK+1 ELSE GOTO 1600
1590 GOSUB 3250
1600 LET MOV=MOV+1:RETURN
1610 REM **PLAYER'S MOVE**
1620 LET X1=0:LET Y1=0
1630 LET PHUFF=0
1640 GOSUB 3800
1650 LET M=0
1660 EVERY 50,1 GOSUB 3200
1670 CLS#2
1680 LOCATE#2,2,2
1690 PRINT#2,"PLAYER'S MOVE"
1700 IF JUMPED=1 THEN LET F$=J$:PRINT#2,"FROM:";F$
:GOTO 1740
1710 INPUT#2,"FROM:",F$
1720 IF LEFT$(F$,1)="P" THEN LET SM=1:GOTO 4540
1730 IF LEN(F$)<>2 THEN GOTO 1670
```

```
1740 INPUT#2,"  TO:",G$
1750 IF LEN(G$)>2 THEN GOTO 1670
1760 IF JUMPED=1 AND G$="" THEN LET JUMPED=0:RETUR
N
1770 IF F$="" OR G$="" THEN GOTO 1870
1780 LET X1=ASC(LEFT$(F$,1))-64
1790 LET X2=ASC(LEFT$(G$,1))-64
1800 LET Y1=ASC(RIGHT$(F$,1))-48
1810 LET Y2=ASC(RIGHT$(G$,1))-48
1820 IF B$(X1,Y1)="1" THEN LET M=1:GOTO 1850
1830 IF B$(X1,Y1)="2" THEN LET M=2:GOTO 1850
1840 GOTO 1870
1850 GOSUB 2000
1860 IF LEGAL=1 THEN GOTO 1900
1870 LOCATE#2,1,5:PRINT#2,"ILLEGAL MOVE"
1880 FOR I=1 TO 1600:NEXT I
1890 GOTO 1670
1900 LET B$(X1,Y1)="0"
1910 LET LX2=X2:LET LY2=Y2:LET LX1=X1:LET LY1=Y1
1920 IF M=1 THEN LET B$(X2,Y2)="1" ELSE LET B$(X2,
Y2)="2"
1930 IF ABS(X2-X1)=2 THEN LET B$((X1+X2)/2,(Y1+Y2)
/2)="0":LET PHUFF=0:LET JUMPED=1:LET J$=G$:LET CC=
CC-1
1940 GOSUB 3250
1950 IF Y2=8 AND B$(X2,Y2)="1" THEN LET B$(X2,Y2)=
"2":LET PK=PK+1:LET JUMPED=0 ELSE GOTO 1970
1960 GOSUB 3250
1970 IF JUMPED =1 THEN GOTO 1610
1980 IF PHUFF=0 THEN RETURN ELSE GOSUB 4000
1990 RETURN
2000 REM **LEGALITY CHECK**
2010 LET LEGAL=1
2020 IF X1<1 OR X1>8 THEN LET LEGAL=0:RETURN
2030 IF X2<1 OR X2>8 THEN LET LEGAL=0:RETURN
2040 IF Y1<1 OR Y1>8 THEN LET LEGAL=0:RETURN
2050 IF Y2<1 OR Y2>8 THEN LET LEGAL=0:RETURN
2060 IF VAL(B$(X1,Y1))<>M THEN LET LEGAL=0:RETURN
2070 IF M=1 AND Y2-Y1<0 THEN LET LEGAL=0:RETURN
2080 IF B$(X2,Y2)<>"0" THEN LET LEGAL=0:RETURN
2090 IF ABS(X2-X1)>2 OR ABS(Y2-Y1)>2 THEN LET LEGA
L=0:RETURN
2100 IF ABS(X2-X1)=0 OR ABS(Y2-Y1)=0 THEN LEGAL=0:
RETURN
2110 IF ABS(X2-X1)<>ABS(Y2-Y1) THEN LET LEGAL=0:RE
TURN
2120 IF M=1 AND ABS(X2-X1)=2 AND VAL(B$((X1+X2)/2,
(Y1+Y2)/2))>=0 THEN LET LEGAL=0:RETURN
2130 IF M=-1 AND ABS(X2-X1)=2 AND VAL(B$((X1+X2)/2
,(Y1+Y2)/2))<=0 THEN LET LEGAL=0:RETURN
2140 IF JUMPED=1 AND ABS(X2-X1)<>2 THEN LET LEGAL=
```

```
0:RETURN
2150 IF M=2 AND ABS(X2-X1)>1 AND VAL(B$((X2+X1)/2,
(Y2+Y1)/2))>=0 THEN LEGAL=0:RETURN
2160 IF M=-2 AND ABS(X2-X1)>1 AND VAL(B$((X2+X1)/2
,(Y2+Y1)/2))<=0 THEN LEGAL=0:RETURN
2170 RETURN
2180 REM **MULTIPLE JUMP SENSE**
2190 LET XM1=X2:LET YM1=Y2
2200 LET X1=X2:LET Y1=Y2
2210 IF M=-1 THEN LET Y2=Y1-2 ELSE FOR Y2=Y1-2 TO
Y1+2 STEP 4
2220 FOR X2=X1-2 TO X1+2 STEP 4
2230 IF X2>8 THEN GOTO 2260
2240 GOSUB 2000
2250 IF LEGAL=1 THEN LET AGAIN=1:RETURN
2260 NEXT X2
2270 IF M=-2 THEN NEXT Y2
2280 LET AGAIN=0:RETURN
2290 REM ** MULTIPLE JUMP EXECUTE**
2300 X1=X2:Y1=Y2
2310 IF M=-1 THEN LET Y2=Y1-2 ELSE FOR Y2=Y1-2 TO
Y1+2 STEP 4
2320 FOR X2=X1-2 TO X1+2 STEP 4
2330 IF X2>8 THEN GOTO 2360
2340 GOSUB 2000
2350 IF LEGAL =1 THEN RETURN
2360 NEXT X2
2370 IF M=-2 THEN NEXT Y2
2380 RETURN
2390 REM **LOOK AHEAD MULTIPLE**
2400 LET XR2=X2:LET YR2=Y2
2410 X1=X2:Y1=Y2
2420 IF M=1 THEN LET Y2=Y1+2 ELSE FOR Y2=Y1+2 TO Y
1-2 STEP -4
2430 FOR X2=X1-2 TO X1+2 STEP 4
2440 IF X2>8 THEN GOTO 2480
2450 GOSUB 2000
2460 IF LEGAL=1 THEN LET AGAIN=1:RETURN
2470 NEXT X2
2480 IF M=2 THEN NEXT Y2
2490 LET AGAIN=0
2500 RETURN
2510 REM **CHECK FOR END OF GAME**
2520 RETURN
2530 REM **LOOK AHEAD SECTION**
2540 LET JUMPING=0
2550 FOR Y1=1 TO 8
2560 FOR X1=1 TO 8
2570 LET M$=B$(X1,Y1):LET M=VAL(M$)
2580 IF M<1 THEN GOTO 2840
2590 FOR X2=X1-2 TO X1+2 STEP 4
```

```
2600 IF X2>8 THEN GOTO 2800
2610 IF B$(X1,Y1)="1" THEN LET Y2=Y1+2 ELSE FOR Y2
=Y1-2 TO Y1+2 STEP 4
2620 GOSUB 2000
2630 IF LEGAL=0 AND M=2 THEN GOTO 2810
2640 IF LEGAL=0 THEN GOTO 2800
2650 IF ABS(X2-X1)=2 THEN LET TES=1:LET XD1=X1:LET
 XD2=X2:LET YD1=Y1:LET YD2=Y2
2660 LET X$=B$((X1+X2)/2,(Y1+Y2)/2)
2670 IF LEGAL=1 AND ABS(X2-X1)=2 THEN LET W=W-5:LE
T B$(X1,Y1)="0":LET B$(X2,Y2)=RIGHT$(STR$(M),1):LE
T B$((X2+X1)/2,(Y1+Y2)/2)="0":LET JUMPING=JUMPING+
1:GOSUB 2390
2680 IF (CC+3)<PC THEN LET W=W-2
2690 IF X$="-2" THEN LET W=W-5
2700 IF AGAIN=1 THEN  GOTO 2660
2710 IF TES=1 THEN GOSUB 3000
2720 IF TES=1 THEN LET TES=0:LET X1=XD1:LET X2=XD2
:LET Y1=YD1: LET Y2=YD2
2730 IF X2=1 AND B$(3,Y2)="-2" THEN LET W=W+1
2740 IF X2=8 AND B$(6,Y2)="-2" THEN LET W=W+1
2750 IF Y2=8 AND B$(X2,6)="-2" THEN LET W=W+1
2760 IF Y2=1 AND B$(X2,3)="-2" THEN LET W=W+1
2770 IF Y2<3 OR Y2>6 OR X2<3 OR X2>6 THEN GOTO 280
0
2780 IF B$(X2,Y2)="2" AND VAL(B$(X2-1,Y2-1))<0 AND
 B$(X2-2,Y2-2)="0" AND VAL(B$(X2+1,Y2+1))<0 AND B$
(X2+2,Y2+2)="0" THEN LET W=W-8
2790 IF B$(X2,Y2)="2" AND VAL(B$(X2-1,Y2+1))<0 AND
 B$(X2-2,Y2+2)="0" AND VAL(B$(X2+1,Y2-1))<0 AND B$
(X2+2,Y2-2)="0" THEN LET W=W-8
2800 FOR I=1 TO 8:FOR J=1 TO 8:LET B$(J,I)=D$(J,I)
:NEXT J,I
2810 LET M=VAL(M$):IF M=2 AND X2<9 THEN NEXT Y2
2820 NEXT X2
2830 IF MJ=1 THEN RETURN
2840 NEXT X1
2850 NEXT Y1
2860 RETURN
2870 REM **CONSIDER ATTACK POSITION**
2880 IF Y2=1 THEN GOTO 2930
2890 IF X2=1 THEN GOTO 2920
2900 IF VAL(B$(X2-1,Y2-1))>0 AND B$(X2-2,Y2-2)="0"
 THEN LET W=W+2
2910 IF X2=8 THEN RETURN
2920 IF VAL(B$(X2+1,Y2-1))>0 AND B$(X2+2,Y2-2)="0"
 THEN LET W=W+2
2930 IF M=-1 THEN RETURN
2940 IF Y2=8 THEN RETURN
2950 IF X2=8 THEN GOTO 2970
2960 IF VAL(B$(X2+1,Y2+1))>0 AND B$(X2+2,Y2+2)="0"
```

```
 THEN LET W=W+2
2970 IF X2=1 THEN RETURN
2980 IF VAL(B$(X2-1,Y2+1))>0 AND B$(X2-2,Y2+2)="0"
 THEN LET W=W+2
2990 RETURN
3000 REM **PLAYER ATTACK POSITION**
3010 LET X2=XR2:LET Y2=YR2
3020  LET VP=VAL(B$(X2,Y2))*5
3030 IF EG>0 AND PC<=CC THEN LET VP=VP+5
3040 IF Y2=1 THEN GOTO 3110
3050 IF X2=1 THEN RETURN
3060 IF X2>=8 THEN RETURN
3070 IF VAL(B$(X2-1,Y2+1))<0 AND B$(X2+1,Y2-1)="0"
 THEN LET W=W+VP:RETURN
3080 IF VAL(B$(X2+1,Y2+1))<0 AND B$(X2-1,Y2-1)="0"
 THEN LET W=W+VP:RETURN
3090 IF Y2=8 THEN RETURN
3100 IF B$(X2+1,Y2-1)<>"-2" THEN GOTO 3120
3110 IF VAL(B$(X2+1,Y2-1))<0 AND B$(X2-1,Y2+1)="0"
 THEN LET W=W+VP:RETURN
3120 IF B$(X2-1,Y2-1)<>"-2" THEN RETURN
3130 IF VAL(B$(X2-1,Y2-1))<0 AND B$(X2+1,Y2+1)="0"
 THEN LET W=W+VP:RETURN
3140 RETURN
3150 REM **COMPUTER'S CLOCK**
3160 LET CSEC=CSEC+1:IF CSEC=60 THEN LET CSEC=0:LE
T CMI=CMI+1
3170 LOCATE#3,3,3:PRINT#3,USING"##";CMI;:PRINT#3,"
:";:PRINT#3,USING"##";CSEC
3180 IF CSEC<10 THEN LOCATE#3,6,3:PRINT#3,"0"
3190 RETURN
3200 REM **PLAYER'S CLOCK**
3210 LET PSEC=PSEC+1:IF PSEC=60 THEN LET PSEC=0:LE
T PMI=PMI+1
3220 LOCATE#4,3,3:PRINT#4,USING"##";PMI;:PRINT#4,"
:";:PRINT#4,USING"##";PSEC
3230 IF PSEC<10 THEN LOCATE#4,6,3:PRINT#4,"0"
3240 RETURN
3250 REM **DRAW THE BOARD**
3260 LET D=REMAIN(1)
3270 AS=1
3280 INK 2,26
3290 INK 3,0
3300 PAP=2
3310 FOR Y=1 TO 8
3320 FOR X=1 TO 8
3330 PAPER PAP
3340 IF B$(X,Y)="1"THEN PEN 1:LOCATE X*2-1,Y*2-1:P
RINT CHR$(251);CHR$(252): LOCATE X*2-1,Y*2:PRINT C
HR$(253);CHR$(254)
3350 IF B$(X,Y)="-1" THEN PEN 0:LOCATE X*2-1,Y*2-1
```

```
:PRINT CHR$(251);CHR$(252):LOCATE X*2-1,Y*2:PRINT
CHR$(253);CHR$(254)
3360 IF B$(X,Y)="0" THEN LOCATE X*2-1,Y*2-1:PRINT"
  ":LOCATE X*2-1,Y*2:PRINT"   "
3370 IF B$(X,Y)="-2" THEN PEN 0:LOCATE X*2-1,Y*2-1
:PRINT CHR$(247);CHR$(248):LOCATE X*2-1,Y*2:PRINT
CHR$(249);CHR$(250)
3380 IF B$(X,Y)="2"THEN PEN 1:LOCATE X*2-1,Y*2-1:P
RINT CHR$(247);CHR$(248): LOCATE X*2-1,Y*2:PRINT C
HR$(249);CHR$(250)
3390 PAP=PAP+1
3400 IF PAP=4 THEN LET PAP=2
3410 NEXT X
3420 IF Y/2<>INT(Y/2) THEN LET PAP=3 ELSE LET PAP=
2
3430 IF Y=1 THEN LET PAP=3
3440 NEXT Y
3450 PLOT 0,400,2
3460 MOVE 10,136:TAG
3470 PRINT"A B C D E F G H";
3480 FOR I=390 TO 160 STEP-32
3490 MOVE 260,I:TAG
3500 PRINT USING"#";AS;
3510 AS=AS+1
3520 NEXT I
3530 TAGOFF
3540 RETURN
3550 REM **SET UP BOARD**
3560 FOR I=1 TO 8
3570 FOR J=1 TO 8
3580 LET B$(I,J)="0"
3590 NEXT J
3600 NEXT I
3610 FOR I=1 TO 8 STEP 2
3620 LET B$(I,1)="1"
3630 LET B$(I,7)="-1"
3640 LET B$(I,3)="1"
3650 NEXT I
3660 FOR I=2 TO 8 STEP 2
3670 LET B$(I,2)="1"
3680 LET B$(I,8)="-1"
3690 LET B$(I,6)="-1"
3700 NEXT I
3710 FOR I=0 TO 9
3720 LET B$(I,0)="*"
3730 LET B$(I,9)="*"
3740 LET B$(0,I)="*"
3750 LET B$(9,I)="*"
3760 NEXT I
3770 RETURN
3780 REM **STALE MATE**
```

```
3790 LET SM=1:GOTO 4540
3800 REM **CHECK TO HUFF PLAYER**
3810 FOR I=1 TO 8
3820 FOR J=1 TO 8
3830 LET HP$(I,J)=B$(I,J)
3840 NEXT J,I
3850 LET PX1=X2:LET PY1=Y2
3860 FOR Y1=1 TO 8
3870 FOR X1=1 TO 8
3880 FOR X2=X1-2 TO X1+2 STEP 4
3890 IF X2>8 THEN GOTO 3980
3900 LET M$=B$(X1,Y1):LET M=VAL(M$)
3910 IF M<=0 THEN GOTO 3980
3920 IF M=1 THEN LET Y2=Y1+2 ELSE FOR Y2=Y1+2 TO Y
1-2 STEP -4
3930 GOSUB 2000
3940 IF LEGAL=1 AND PHUFF=0 THEN LET PHUFF=1:RETUR
N
3950 IF LEGAL=1 AND (X1<>PX1 AND Y1<>PY1) THEN LET
 PHUFF=1:RETURN
3960 IF M=2 THEN NEXT Y2
3970 NEXT X2
3980 NEXT X1,Y1
3990 LET PHUFF=0:RETURN
4000 REM **HUFF PLAYER**
4010 FOR J=1 TO 8
4020 FOR I=1 TO 8
4030 LET HS$(I,J)=B$(I,J)
4040 LET B$(I,J)=HP$(I,J)
4050 NEXT I,J
4060 GOSUB 3860
4070 FOR J=1 TO 8
4080 FOR I=1 TO 8
4090 LET B$(I,J)=HS$(I,J)
4100 NEXT I,J
4110 IF X1=LX1 AND Y1=LY1 THEN LET X1=LX2:LET Y1=L
Y2
4120 PRINT#2,"HUFF YOU AT ";CHR$(X1+64);CHR$(Y1+48
)
4130 LET PC=PC-1
4140 LET B$(X1,Y1)="0":GOSUB 3250
4150 RETURN
4160 REM **KING MOVEMENT**
4170 IF PC>2 OR CC-CK>0 THEN RETURN
4180 FOR J=1 TO 8
4190 FOR I=1 TO 8
4200 IF B$(I,J)="2" THEN GOTO 4220
4210 NEXT I,J
4220 LET XP1=I:LET YP1=J
4230 LET CBD=ABS(XP1-X1M)+ABS(YP1-Y1M)
4240 LET W=W-(14-CBD)
```

```
4250 IF ABS(XP1-X2M)>ABS(XP1-X1M) THEN LET W=W-2
4260 IF ABS(YP1-Y2M)>ABS(YP1-Y1M) THEN LET W=W-2
4270 RETURN
4280 REM **END GAME ROUTINE**
4290 LET TC=0:LET TP=0
4300 FOR I=1 TO 8
4310 FOR J=1 TO 8
4320 IF VAL(B$(J,I))>1 THEN LET TP=TP+I
4330 IF VAL(B$(J,I))<-1 THEN LET TC=TC+I
4340 NEXT J,I
4350 LET TC=TC/CC
4360 LET TP=TP/PC
4370 IF TP>TC THEN LET EG=1 ELSE LET EG=2
4380 RETURN
4390 REM **CHECK FOR REPETITION**
4400 FOR I=1 TO 8
4410 FOR J=1 TO 8
4420 IF B$(I,J)<>GH$(I,J) THEN GOTO 4460
4430 NEXT J
4440 NEXT I
4450 LET REP=REP+1:IF REP=4 THEN LET REP=0:LET REP
EAT=1:RETURN
4460 LET REP=0
4470 LET REPEAT=0
4480 FOR I=1 TO 8
4490 FOR J=1 TO 8
4500 LET GH$(I,J)=B$(I,J)
4510 NEXT J
4520 NEXT I
4530 RETURN
4540 REM **END GAME ROUTINE**
4550 CLS#2
4560 PRINT#2,"     GAME OVER"
4570 PRINT#2
4580 IF SM=1 THEN GOTO 4630
4590 PRINT#2,"THE ";Z$;" WINS"
4600 PRINT#2
4610 INPUT#2,"PRESS ENTER FOR NEWGAME";Z$
4620 RUN
4630 PRINT #2,"   STALE MATE"
4640 PRINT#2
4650 INPUT#2,"PRESS ENTER FOR NEWGAME";Z$
4660 RUN
```

12. Connect Four

Summary

So far this book has dealt with the uses of artificial intelligence and some of the major extensions to Basic which Locomotive Basic offers, developing the use of both in a selection of 'intelligent' games and puzzles. In the puzzles, the machine displays a level of artificial intelligence by forbidding you to make illegal moves, which it achieves by comparing the move you have requested against an inbuilt criterion of legality. For example, in the Towers of Hanoi puzzle you cannot place a larger disc on one of smaller size, and in the Sliding Puzzle you are only permitted to slide an adjacent tile into the empty space. Without these rules the puzzles would often become trivial to solve and thereby offer no intellectual stimulus, which is the aim of a computer challenge.

For the games, however, an extension of the above intelligence is required, since, although confirmation of legality is an important and vital step, the computer must also decide on a move which will make it as difficult as possible for you to obtain a victory. This additional display of artificial intelligence can be obtained in several different ways, and a selection has been employed in the games in this book to facilitate sensible play by the computer.

In Noughts and Crosses, if the computer goes first it adopts an algorithmic approach which develops after each square of the grid has been occupied. This is achieved by selecting a square at random (either 1,3,5,7 or 9) and then playing a different game depending on this choice. Should it be required to defend, the computer employs a heuristic approach as it has been educated in many of the tricks which can be used to obtain a victory and is therefore able to block such attempts at winning the game.

With Cribbage, the computer considers in turn the fifteen different ways of choosing four of the six cards it has been dealt, and by checking to see the score for each hand it is able to select the one with the maximum number of points. By using this method the computer never gambles on the cut of the cards which the human player might attempt, but is instead content to play a safe game which will probably prove to be successful in the

long term. An extra artificial intelligence routine is required in this program since the game also comprises the 'play for points' section, where the players lay their cards alternately in an attempt to reach 31, obtain pairs or runs, etc., and the computer must be able to perform this intelligently if it is to have a chance of winning.

The third technique employed is that of weighting, in which each legal move is considered and the resulting position is weighted on its benefit to either player, i.e. the higher the weighting, the better the position for the Amstrad. This is particularly useful with board games when it is not clear what the best move may be and an algorithm for victory cannot be produced. Clearly the computer will play better if the position after several subsequent moves is examined, but this is extremely time-consuming and is therefore not possible when writing in Basic. However, a compromise is reached in the game of Draughts, in which a deeper level of look-ahead is employed on some occasions, notably when there are fewer pieces on the board and therefore fewer legal possibilities to consider.

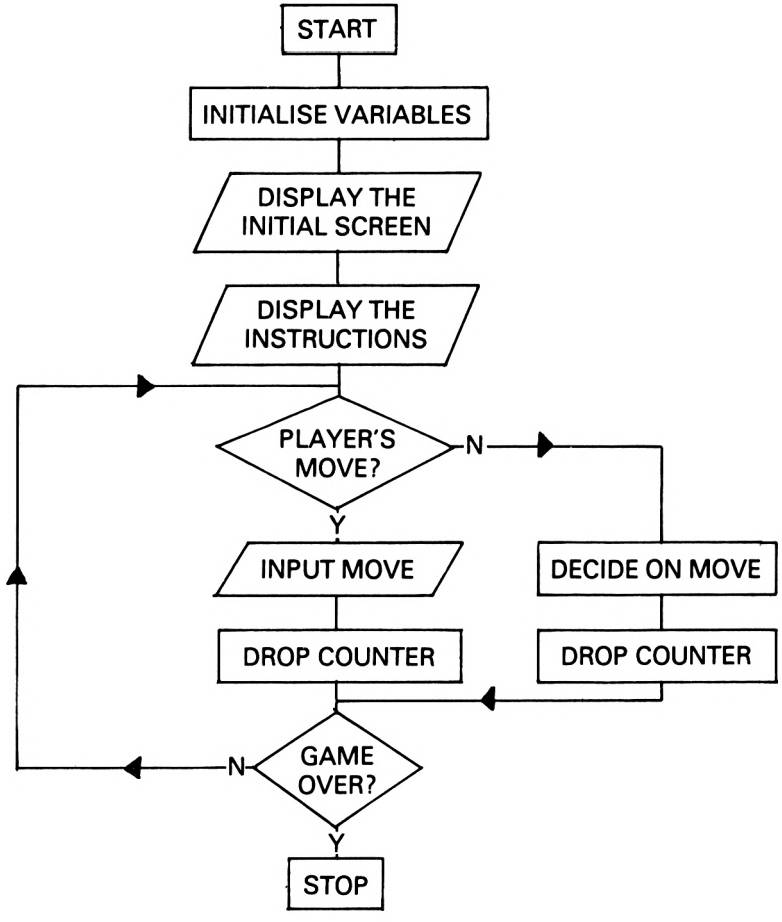
This book is meant to be instructive as well as entertaining. With this in mind, the remaining pages are used to demonstrate the development of the game Connect Four from the initial conception up to the time of writing the routine for the computer's move. This step is omitted to allow you the opportunity of writing one of the artificial intelligence routines of the game. Some hints on the route to take are included.

Connect Four

The game is played on a vertical board with seven columns and six rows of holes which can be filled by a counter. You and the computer take alternate turns at dropping a counter into one of the columns with the aim of obtaining a row of four either vertically, horizontally or diagonally.

The first step in writing the program is to construct a simple flowchart outlining the basic blocks which will need to be formulated separately.

Let us consider each step individually, constructing the code in the form of subroutines.



Initialise variables

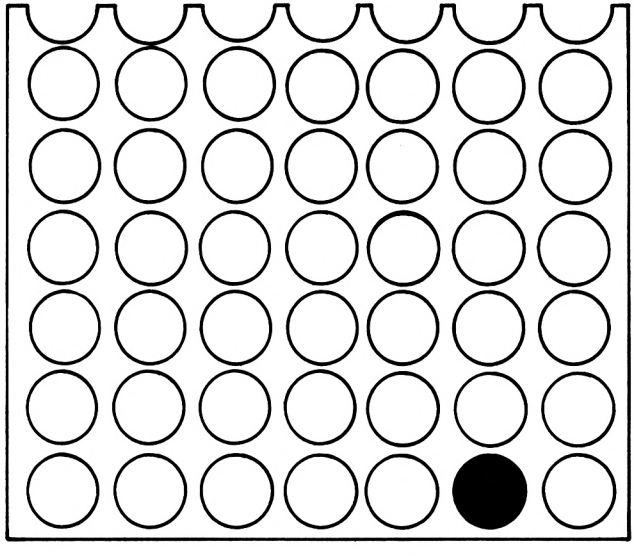
The program is to be written in MODE 1, and so this must be selected and the colours to be used must be defined. The board is to be black with white holes and the counters are to be red and yellow, so the colours are defined as follows:

0 white 1 black
2 red 3 yellow

Next we need a variable to represent the current state of the board, and the best way of achieving this is to use a

two-dimensional array B(x,y) (see below). The contents of the array are given the value 0 at the start of the game, and this value is subsequently altered when occupied by a counter.

The other major requirement is to create the user-defined graphics which are to represent the disc, and this is constructed in four characters in the configuration of a square.



B(6,1) is coloured.

The subroutine which performs this initialisation is contained in lines 1000-1110.

```
1000 REM ** INITIALIZE VARIABLES **
1010 MODE 1
1020 INK 0,26:INK 1,0:INK 2,6:INK 3,24
1030 BORDER 18
1040 DIM B(7,6)
1050 FOR I=1 TO 7:FOR J=1 TO 6:LET B(I,J)=0:NEXT J
,I
1060 SYMBOL AFTER 240
1070 SYMBOL 240,&3,&F,&1F,&3F,&7F,&7F,&FF,&FF
1080 SYMBOL 241,&C0,&F0,&F8,&FC,&FE,&FE,&FF,&FF
1090 SYMBOL 242,&FF,&FF,&7F,&7F,&3F,&1F,&F,&3
1100 SYMBOL 243,&FF,&FF,&FE,&FE,&FC,&F8,&F0,&C0
1110 RETURN
```

Initial screen display

The subroutine which constructs this is to be written in lines 1500-1580. The board is defined as a window (WINDOW #1) and the paper colour selected is black, with the window being cleared to this colour. To display the holes as shown in the diagram above another subroutine is used, and to make this more useful, it is dependent on some variables which must be given values before the routine is called.

It requires (X,Y), the co-ordinates in window #1 of the top left quadrant of the circle, and a value for COL which is to be the colour of the circle it produces. This routine is to be found in lines 2000–2040. A similar routine to construct the half circles at the top of the board is also required, and that is found in lines 2100–2130.

```
1500 REM ** INITIAL SCREEN **
1510 WINDOW #1,3,24,4,23:PAPER #1,1:CLS #1
1520 FOR X=2 TO 20 STEP 3:GOSUB 2100:NEXT X
1530 FOR Y=3 TO 18 STEP 3
1540 FOR X=2 TO 20 STEP 3
1550 COL=0:GOSUB 2000
1560 NEXT X
1570 NEXT Y
1580 RETURN
2000 REM ** DRAW CIRCLE **
2010 PEN #1,COL
2020 LOCATE #1,X,Y:PRINT #1,CHR$(240);CHR$(241)
2030 LOCATE #1,X,Y+1:PRINT #1,CHR$(242);CHR$(243)
2040 RETURN
2100 REM ** DRAW HALF CIRCLES **
2110 PEN #1,0
2120 LOCATE #1,X,1:PRINT #1,CHR$(242);CHR$(243)
2130 RETURN
```

Instructions

The instructions are the next requirement, and these use a second window, displaying the information beside the board. The routine, which also includes the decision on who is to have the first move, is purely textual and is normally left as the last section of the program to be written.

```
7000 REM ** INSTRUCTIONS WINDOW **
7010 WINDOW #2,27,38,3,14:PAPER #2,0:PEN #2,1:CLS
#2
7020 PLOT 414,30,1:DRAW 414,368:DRAW 608,368:DRAW
608,30:DRAW 414,30
```

```
7030 PRINT #2:PRINT #2," CONNECT  4"
7040 PRINT #2
7050 PRINT #2,"   DO YOU":PRINT #2,"  WANT THE":PR
INT #2," FIRST MOVE"
7060 LET FM$=INKEY$
7070 IF FM$<>"Y" AND FM$<>"N" THEN GOTO 7060
7080 CLS #2:PRINT #2:PRINT #2," CONNECT  4":PRINT
#2
7090 PRINT #2,"  USE  THE":PRINT #2,"   CURSOR":PR
INT #2,"  KEYS  TO":PRINT #2," MOVE  YOUR":PRINT #
2,"  COUNTER"
7100 PRINT #2," AND  PRESS":PRINT #2," 'ENTER' TO"
:PRINT #2," RELEASE IT"
7110 WINDOW #3,27,38,15,23:PAPER #3,1:PEN #3,3:CLS
 #3:PRINT #3:PRINT #3,"  YOU  ARE  ";
7120 LOCATE #3,6,3:PRINT #3,CHR$(240);CHR$(241)
7130 LOCATE #3,6,4:PRINT #3,CHR$(242);CHR$(243)
7140 PEN #3,2:PRINT #3:PRINT #3," AMSTRAD IS"
7150 LOCATE #3,6,7:PRINT #3,CHR$(240);CHR$(241)
7160 LOCATE #3,6,8:PRINT #3,CHR$(242);CHR$(243)
7170 IF FM$="Y" THEN GOTO 50
7180 GOTO 70
```

Player's turn

In this section, one of the player's discs is shown at the top of the board and by using the cursor keys it can be moved to the left or to the right. When the disc is in the required position it can then be dropped by pressing the ENTER key.

```
3000 REM ** PLAYER'S TURN **
3010 LET COL=3
3020 LET X1=4:GOSUB 2200
3030 LET M$=INKEY$
3040 IF M$=CHR$(243) THEN GOSUB 3100
3050 IF M$=CHR$(242) THEN GOSUB 3200
3060 IF M$=CHR$(13) THEN GOTO 3300
3070 GOTO 3030
3100 REM ** MOVE RIGHT **
3110 IF X1=22 THEN RETURN
3120 LET X=X1-2:GOSUB 2100
3130 PAPER 0:LOCATE X1,3:PRINT "  "
3140 LET X1=X1+3
3150 GOSUB 2200
3160 RETURN
3200 REM ** MOVE LEFT **
3210 IF X1=4 THEN RETURN
3220 LET X=X1-2:GOSUB 2100
3230 PAPER 0:LOCATE X1,3:PRINT "  "
3240 LET X1=X1-3
```

```
3250 GOSUB 2200
3260 RETURN
3300 REM ** DROP DISC **
3310 LET X=X1-2:LET Y=3:LET I=(X+1)/3:LET J=6
3320 IF B(I,J)<>0 THEN GOTO 3030
3330 GOSUB 9000
3340 GOSUB 2100
3350 PAPER 0:LOCATE X1,3:PRINT "  "
3360 LET COL=3:GOSUB 2000
3370 IF J=1 THEN LET B(I,J)=1:RETURN
3380 IF B(I,J-1)<>0 THEN LET B(I,J)=1:RETURN
3390 GOSUB 9000
3400 LET COL=0:GOSUB 2000
3410 LET J=J-1:LET Y=Y+3
3420 GOTO 3360
```

Another subroutine to draw the disc at the top of the board is required, and this is positioned with other 'disc-drawing' routines at lines 2200–2250.

```
2200 REM ** DISC BEFORE DROP **
2210 PAPER 0:PEN COL
2220 LOCATE X1,3:PRINT CHR$(240);CHR$(241)
2230 PAPER 1:PEN COL
2240 LOCATE X1,4:PRINT CHR$(242);CHR$(243)
2250 RETURN
```

In lines 3000–3070 the keyboard is examined to see which key has been pressed. If it is the right arrow (→) then the routine at 3100 is selected to move the counter to the right, and if it is the left arrow (←) a similar routine for movement is located at line 3200. Both these routines return control to the above section of program. If, however, the ENTER key is pressed, control passes to line 3300 where the disc is dropped.

The animation is created by drawing the disc in the new position and removing it from the old, but because the computer can do this extremely quickly, a time delay is required, and this is in the form of a short subroutine in lines 9000–9020:

```
9000 REM ** PAUSE **
9010 FOR Z=1 TO 1000:NEXT Z
9020 RETURN
```

The final step of this section of code is to set the value of B(x,y) for the final resting place of the disc to 1 and then return to the main program. This occurs in either line 3370 or 3380.

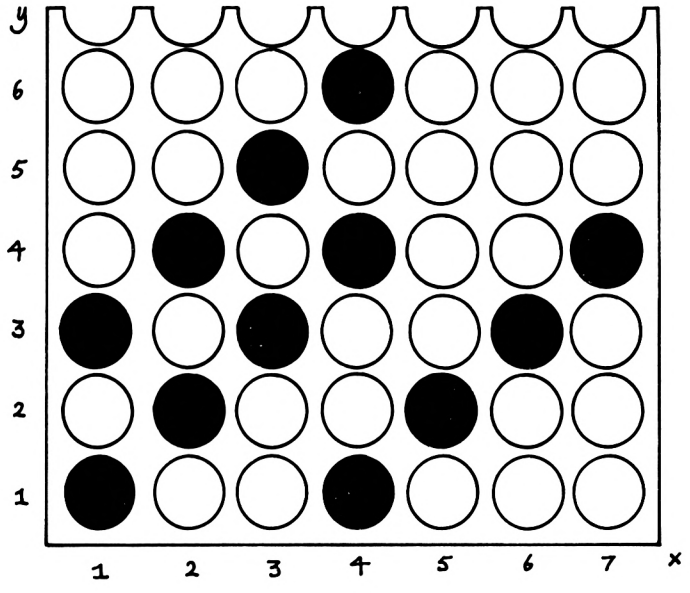
Check for result

This is one of the artificial intelligence routines in the program which checks all the possible winning lines to see if one of the players has won the game. All the individual sections work in a similar way and so we shall only consider one as an example: lines 6140-6190.

This section of program looks for diagonals with a positive gradient by starting at the bottom of the line and checking upwards. The possible values for x are 1,2,3 or 4 and the possible values for y are 1,2 or 3, since any other values would result in the line being off the grid (see diagram below).

The check is related to the starting position by this line of code:

```
6170 IF B(x,y)=A and B(x+1,y+1)=A and B(x+2,y+2)=A and
B(x+3,y+3)=A THEN GOTO 6500
```



which examines the diagonal to the right and upwards of this position. If there is a winning line, the routine at line 6500 is used to display the winner and end the program. If there is no winning line, before control passes to the next player's turn a simple check is made to see if the game is a draw and all of the positions are occupied. However, we do not need to check the whole board,

since a look at the top row will indicate if the match is drawn or not.

```
6000 REM ** CHECK FOR RESULT **
6010 REM ** VERTICAL **
6020 FOR A=-1 TO 1 STEP 2
6030 FOR I=1 TO 7
6040 FOR J=1 TO 3
6050 IF B(I,J)=A AND B(I,J+1)=A AND B(I,J+2)=A AND
 B(I,J+3)=A THEN GOTO 6500
6060 NEXT J
6070 NEXT I
6080 REM ** HORIZONTAL **
6090 FOR J=1 TO 6
6100 FOR I=1 TO 4
6110 IF B(I,J)=A AND B(I+1,J)=A AND B(I+2,J)=A AND
 B(I+3,J)=A THEN GOTO 6500
6120 NEXT I
6130 NEXT J
6140 REM ** DIAGONAL / **
6150 FOR I=1 TO 4
6160 FOR J=1 TO 3
6170 IF B(I,J)=A AND B(I+1,J+1)=A AND B(I+2,J+2)=A
 AND B(I+3,J+3)=A THEN GOTO 6500
6180 NEXT J
6190 NEXT I
6200 REM ** DIAGONAL \ **
6210 FOR I=1 TO 4
6220 FOR J=4 TO 6
6230 IF B(I,J)=A AND B(I+1,J-1)=A AND B(I+2,J-2)=A
 AND B(I+3,J-3)=A THEN GOTO 6500
6240 NEXT J
6250 NEXT I
6260 NEXT A
6270 REM ** MATCH DRAWN **
6280 FOR I=1 TO 7
6290 IF B(I,6)=0 THEN RETURN
6300 NEXT I
6310 CLS #2
6320 PRINT #2:PRINT #2:PRINT #2," GAME DRAWN"
6330 LOCATE 1,1:END
6500 REM ** GAME OVER **
6510 PAPER #2,1:CLS #2
6520 IF A=1 THEN LET Z$="PLAYER ":PEN #2,3
6530 IF A=-1 THEN LET Z$="AMSTRAD":PEN #2,2
6540 PRINT #2:PRINT #2:PRINT #2," GAME  OVER"
6550 PRINT #2:PRINT #2," ";Z$;" IS":PRINT #2," THE
 WINNER"
6560 LOCATE 1,1:END
```

The main program

All the routines are now entered except for the main program and the computer's move. The main program is simply a series of subroutine calls and is extremely short:

```
10 REM ** MAIN PROGRAM **
20 GOSUB 1000
30 GOSUB 1500
40 GOTO 7000
50 GOSUB 3000
60 GOSUB 6000
70 GOSUB 4000
80 GOSUB 6000
90 GOTO 50
```

The computer's move

The routine for the computer's move is complete except for the decision, which is at present random.

```
4000 REM ** COMPUTER'S TURN **
4010 LET COL=2
4020 LET X1=4:GOSUB 2200
4030 LET I=1+INT(7*RND(1))
4040 IF B(I,6)<>0 THEN GOTO 4030
4800 REM ** COMPUTER MOVE TO COLUMN **
4810 FOR P=2 TO I
4820 FOR Z=1 TO 100:NEXT Z
4830 LET X1=1+P*3
4840 LET X=X1-5:GOSUB 2100
4850 PAPER 0:LOCATE X1-3,3:PRINT "  "
4860 GOSUB 2200
4870 NEXT P
4900 REM ** COMPUTER DROP **
4910 LET X=3*I-1:LET Y=3:LET J=6
4920 GOSUB 9000
4930 GOSUB 2100
4940 PAPER 0:LOCATE X1,3:PRINT "  "
4950 LET COL=2:GOSUB 2000
4960 IF J=1 THEN LET B(I,J)=-1:RETURN
4970 IF B(I,J-1)<>0 THEN LET B(I,J)=-1:RETURN
4980 GOSUB 9000
4990 LET COL=0:GOSUB 2000
5000 LET J=J-1:LET Y=Y+3
5010 GOTO 4950
```

The lines at 4030 to 4040 can be removed and then replaced with the artificial intelligence routine which you have designed

from 4030 to 4799. An improvement over a simply random move is to examine the board first and see if three discs are positioned so as to allow another to be added to construct a straight line of four. This can be achieved by looking at all possible straight line combinations and checking the total of the contents of B(x,y) for each of the positions in the line. If this total is −3 then there are three computer discs and one empty position in the line under consideration and code similar to:

```
IF B(x,y)+(B(x+1,y+1)+B(x+2,y+2)+B(x+3,y+3)=−3 THEN ...
```

can be used to redirect the flow to a routine which will isolate the empty position and select the appropriate column from which the counter should be released.

The next improvement would be to design a routine which will block a corresponding counter configuration which the player might have, and this could be done in a similar way to the above, by looking for a total of 3. Further improvement could then be made by attempting to obtain a row of three counters, because from this a winning row of four can be obtained, and improvements can continue to be made by following this pattern of development.

Do not forget, however, that the lines 4030 and 4040 must be included at the end of the routine in case there is no possibility of an intelligent move in which case a random one must be selected.

Variables

In conclusion, here is a table explaining the function of each variable and the values it might take.

B(x,y)	Current board status Any position can hold the value 0,−1 or 1: 0 if the position is unoccupied, −1 if a computer's disc is present and 1 if there is a player's disc there.
X,Y	Co-ordinates in WINDOW #1 of the top left of the disc
X1,Y1	Co-ordinates of the disc before entering WINDOW #1
I	Column number (1-7)
J	Row number (1-6)
A	Player under consideration for victory (1 or −1)
COL	Colour of disc to be drawn (0,1,2 or 3)
FM$	First move (Y or N)

Glossary

Algorithm	A solution to a problem which has been broken down into a logical sequence of steps. A typical example of an algorithm is a flowchart.
Boolean	A boolean variable is a variable which will only hold one of two values, typically 'true' or 'false'.
Buffer	A section of memory which is used to store data when transfer between the computer and a peripheral is taking place. This is required because peripherals often work at a much slower speed to the computer and the data can be held in the buffer while the computer continues with other jobs.
CPU	Central Processing Unit. This is the heart of the computer, in which all the mathematical operations take place.
Cursor	This is a pointer which indicates where the next output will be displayed on the screen. There are two cursors: the text cursor indicates where the text is to be positioned and the graphics cursor is used to indicate where graphics will be displayed.
Database	A large file.
Delimiter	This is a dummy item of data used to separate consecutive pieces of information. A typical use is to indicate the separation of records in a file.
Expert system	This is the name given to a sophisticated file-handling program which uses knowledge as opposed to data to solve complex problems. Currently these represent the major use of artificial intelligence.
Field	The smallest division of a file.
File	A collection of items of data which have something in common.
Flag	A variable which is assigned a certain value when a particular situation arises. It can then be tested throughout the program to indicate the current status.

Global paper	A term introduced in this book to represent INK 0, which is the background colour for all text printed at the graphics cursor.
Heuristic	A set of instructions which cannot guarantee a solution but which, when applied, will lead you closer to one.
Node	The end of a branch of a tree (q.v.).
Pattern recognition	A technique of artificial intelligence which looks for certain standard configurations, e.g. good positions in a board game.
Peripheral	A device external to the computer, e.g. a printer, disc drive, cassette player, etc.
Pixel	Short for picture element. An individually addressable point on the screen.
Queue	A data structure in which the individual items are accessed in the same order as they were stored.
RAM	Random Access Memory. The area of computer memory which is totally accessable to the user.
Record	A subdivision of a file containing a number of closely related fields.
Root	The starting point of a tree (q.v.).
Tree	A data structure which displays a hierarchy in its representation of the information.

Duckworth Home Computing

COMPUTER CHALLENGES FOR THE AMSTRAD
Richard Hurley & David Virgo

With the aid of ten superb programs, this book demonstrates the use of artificial intelligence on the Amstrad CPC464. The first two chapters introduce you to the principles of artificial intelligence and the more advanced features of Locomotive Basic which are used in this book. The rest of the book is divided into two parts: the first contains puzzles for you to solve; and the second a collection of stimulating games in which you will find the computer a worthy adversary.

The puzzles include Crossword Puzzler, which will provide you with an endless supply of crosswords, and The Cube, which is a graphical representation of Rubik's Cube. The games include Cribbage, which will tax your card-playing skills to the limit, Backgammon, complete with on-screen prompts, and Draughts, which is designed to play the best possible game while keeping the time taken for each down to well below one minute.

Richard Hurley is Head of Computer Studies at Hurstpierpoint College in Sussex, and has written several books on computing. David Virgo also teaches computing at Hurstpierpoint.

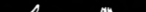
Duckworth
The Old Piano Factory
43 Gloucester Crescent, London NW1

ISBN 0 7156 1979 9

IN UK ONLY £6.95 NET

COMPUTER CHALLENGES FOR THE AMSTRAD

Richard Hurley David Virgo



AMSTRAD CPC

MÉMOIRE ÉCRITE
MEMORY ENGRAVED
MEMORIA ESCRITA

https://acpc.me/