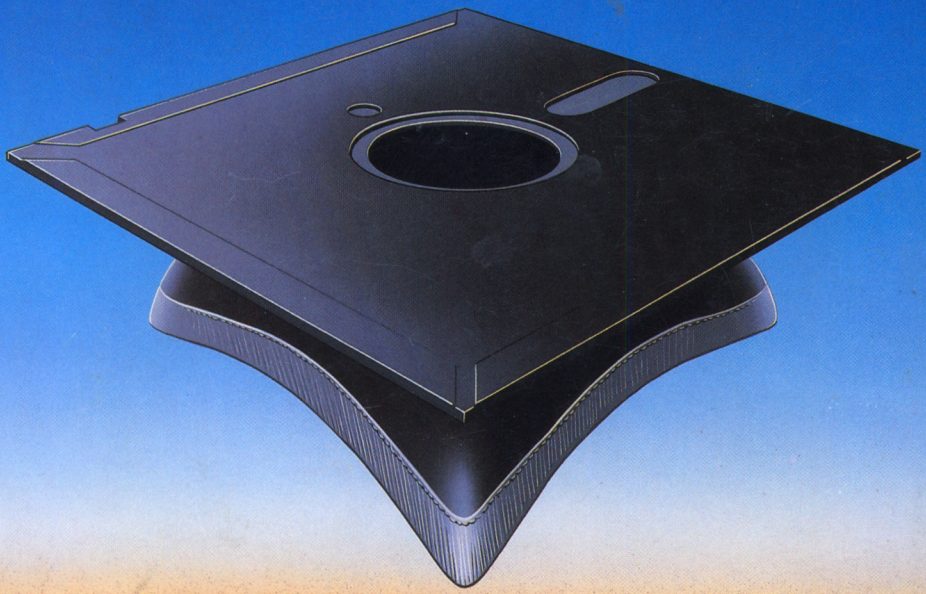
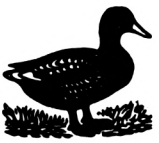
AN EDUCATIONAL DATABASE FOR THE AMSTRAD



Richard Hurley & David Virgo

An Educational Database 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 1977 2

British Library Cataloguing in Publication Data

Hurley, Richard G.
An educational database for the Amstrad.
1. Computer-assisted instruction 2. Amstrad CPC464 (Computer)—Programming
I. Title II. Virgo, David
371.3'9445 LB1028.5

ISBN 0-7156-1977-2

Photoset in North Wales by
Derek Doyle & Associates, Mold, Clwyd.
Printed in Great Britain by
Biddles Ltd., Guildford and King's Lynn

Contents

	Preface	7
	Introduction	9
1.	Databases	11
2.	Files on the Amstrad	14
3.	The Database (EDBASE)	23
4.	Type Attack	44
5.	Crossword Puzzler	50
6.	Wordsearch	65
7.	Horseracing	77
8.	Nightmare Park	85
9.	The Card Player	96
10.	Hangman	111
11.	The Geographer	120
12.	Adders and Ladders	149
13.	Blockbusters	162
14.	Wordsquare	174
15.	Pick a Pair	183
	Appendix	192

This book is dedicated to Stephen
for all his help with this and other projects.

Preface

Computers now play an important part in children's education. We live in a highly technological world full of computers and electronics, and there are few areas of life which do not make use of them. Home computers have been around since 1980, when Clive Sinclair introduced his £100 ZX-80 computer. Since then the number of machines available and their overall sophistication has vastly increased. Today's home computers have more memory, work faster and are able to produce brilliant graphical displays.

This is a book of educational programs aimed at children from the age of nine upwards, but there is no reason why adults should not join in and play some of the games. There are detailed notes on the construction of the programs, along with flowcharts for every program, which together will give both children and parents an insight into their operation. No prior knowledge of programming is required beyond the ability to type with one finger.

In this book we make use of the facilities of the Amstrad CPC464 computer to produce a series of programs which can be used to educate as well as to demonstrate the power of the machine. Most of the programs make use of an educational database, introduced in Chapter 3, but we have also included a few mathematical programs which do not rely on the database.

R.H. & D.V.

Introduction

From a technical point of view this book employs the principles and techniques of Database Management to provide the user with a number of CAL (Computer Assisted Learning) programs. Each of these programs is aimed at children between the ages of nine and fifteen, and together they can be employed to teach general knowledge, vocabulary and geography as well as the basic rules of arithmetic.

Learning subjects such as vocabulary and arithmetic can be very tedious, and the programs contained in this text are designed to make this process as entertaining and interesting as possible by using a series of sophisticated games and puzzles, each designed by a qualified teacher with several years of experience. Most of the programs make use of a central database containing words and facts on a number of subjects. Detailed information is given throughout the book on how this collection of knowledge can be expanded, thus increasing the sophistication of the games and extending their usefulness.

This is not the first, nor is it likely to be the last book of educational programs. However, unlike many of its predecessors it does not contain a large number of trivial programs but a small collection of complex listings, which by use of the database can be extended as the needs of the child change.

Each listing in this book is preceded by a warning such as that show below. It must be strictly adhered to if the programs are to function correctly.

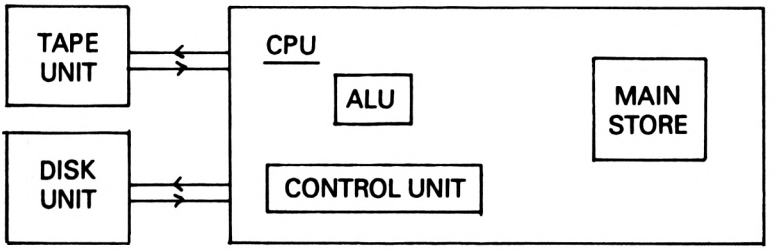
The game should be entered and played in the caps lock mode.
All spaces within the text should be entered as they appear.

We recommend that you make at least two copies of each program on a good quality cassette in case of accidental erasure or a faulty saving process. We also strongly advise you not to overwrite old copies of the database when you have amended it and are about to make new copies; keep them as a reserve should any errors occur.

If, when the games have been entered and saved on to tape, you wish to use several of the programs one after the other, then it is a good idea to reset the machine at the end of each program so that the colours and inks are returned to their normal state.

1. Databases

For the majority of computer applications, vast quantities of data are required. This is then processed by using a program to give the required results. In most practical examples information is stored in the form of 'text files' on some kind of backing store such as magnetic tape or disks and then transferred into the computer's main store as and when required.



The diagram shows the layout of a typical computer system containing two peripherals on which the files can be stored. Each unit may be capable of storing many times the total capacity of the main store. Therefore, by transferring information at the appropriate time, the computer user has access to a far greater amount of data.

The major drawback of these peripheral units is their slow speed of operation in comparison with the central processing unit, or CPU, which may be capable of working many thousands of times faster. From an electronic or hardware viewpoint the problem is overcome by using an internal buffer into which information is temporarily placed. However, from the user's point of view the slowness of transferring information from the backing store to the CPU and vice-versa can only be overcome by using the fastest possible peripherals and some sophisticated programming techniques.

Magnetic tape

If magnetic tape (cassette) is used, the problem of time delay is at its most pronounced. Tapes work very slowly, and the information must be stored in a serial (one after the other) format.

In most cases in which a tape is used, the whole file will be read into memory before any processing is started, a process used throughout this book. This method is satisfactory but limits the size of file which can be used, as the main store must contain the file as well as the program.

Magnetic disks

Unlike tape, magnetic disks work at a much higher speed and are able to store individual pieces of information in addressable positions on the surface of the disk in a way similar to that in which data is stored within the computer. Because of this it is not necessary to read the complete file; records can be located and transferred as and when required. This process is known as random access, and by constructing a file using this technique information can be obtained very quickly indeed.

Defining a Database

Although this chapter is entitled 'Databases', we keep referring to files. We should now attempt to define the difference between these two terms.

A database is a very large text file containing information which can be used by a number of different programs for different purposes, with each program extracting only the required information from the database. This is different from a simple file, which is likely to be very program-dependent and contain only a limited quantity of information.

The difference between a database and a file is best viewed in the context of the way in which the programs are written:

1. In the case of a file, it is likely that the program will be written first and then the file constructed to contain the relevant information.

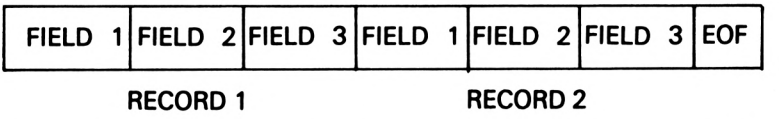
2. For a database, however, programs would be designed after the database was created (as in this book), using some or all of the information it contains.

2. Files on the Amstrad

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 chapter we will consider how this can be achieved using an Amstrad computer and the Datacorder to store the information on tape.

In data processing, the quantity of information being considered is often very large. It is therefore impractical to store all this information in data statements within the program. Under such circumstances, the data would be stored in a file which would 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, 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'

It is important to know the type of each field so that the Basic instruction can use the correct type of variable. For example, if the field is numeric then an ordinary variable such as 'A' can be used,

otherwise a string variable 'A$' is necessary.

Key field

In general, the first field of each record is called the key field. This is used to identify a particular record, distinguishing 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 which the file contains. If under these circumstances we attempt to read the file by using a FOR-NEXT loop, 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 relating to cassette file handling on the Amstrad. They will be described in this chapter and used throughout the book.

```
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 it is either 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 (I/O) 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. This can be seen during the execution of many of the programs in this book.

PRINT#9
The PRINT#9 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 command is the opposite of the PRINT#9 command, transferring one item of data (field) from the file into the main store via the I/O 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.

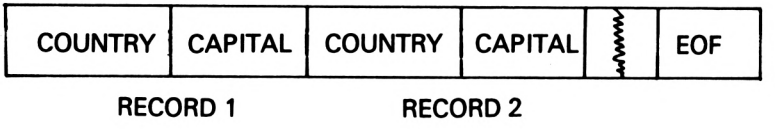
Example 1
The following program constructs a file called CAPITAL, which contains ten countries and their respective capitals with the original information being obtained from data.

```
10 REM **FILE CREATE**
20 CLS
30 DIM A$(10),B$(10)
40 FOR I=1 TO 10
50 READ A$(I),B$(I)
60 NEXT I
70 OPENOUT "CAPITAL"
80 FOR I=1 TO 10
90 PRINT#9,A$(I)
100 PRINT#9,B$(I)
110 NEXT I
120 CLOSEOUT
130 END
140 DATA FRANCE,PARIS,GREECE,ATHENS,ITALY,ROME,SPA
IN,MADRID,PORTUGAL,LISBON,GERMANY,BERLIN,RUSSIA,MO
SCOW,BRITAIN,LONDON,AMERICA,WASHINGTON,JAPAN,TOKYO
```

Comments

Lines 10-60	Initialise variables
Line 70	Open file
Lines 80-110	Transfer records from memory to tape
Lines 120-130	Close file and end program
Line 140	Data for file

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



Read only

Having created a file on tape using a write only program, as demonstrated 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 such as that shown below.

Example 2
The following program reads the CAPITAL file created in Example 1 and prints the various countries and capitals on the screen in the form of a table.

```
10 CLS
20 DIM A$(20),B$(20)
30 LET I=1
40 OPENIN "CAPITAL"
50 INPUT#9,A$(I)
60 INPUT#9,B$(I)
70 LET I=I+1
80 IF NOT EOF THEN GOTO 50
90 CLOSEIN
100 PRINT "COUNTRY","CAPITAL"
110 PRINT "=======","======="
120 FOR J=1 TO I-1
130 PRINT A$(J),B$(J)
140 NEXT J
150 END
```

Comments

Lines 10-20	Initialise variables
Line 30	Set initial value for array
Lines 40-90	Transfer information from file into RAM
Lines 100-150	Print information on the screen

In this example the whole of the file is read into the computer. However, on occasion this might not be possible or even necessary as we may only wish to interrogate the file to find one piece of information.

Example 3
The following program allows the user to input a country. The computer then interrogates the file in order to find the corresponding capital.

```
10 CLS
20 INPUT "COUNTRY=";A$
30 OPENIN "CAPITAL"
40 INPUT#9,B$
50 INPUT#9,C$
60 IF A$=B$ THEN GOTO 90
70 IF EOF THEN GOTO 130
80 GOTO 40
90 CLOSEIN
100 PRINT "CAPITAL=";C$
110 END
120 CLOSEIN
130 PRINT "COUNTRY NOT ON FILE"
140 END
```

Comments

Line	20	User inputs required country
Line	30	Open file
Lines	40-60	Input record and check against A$
Line	70	Check for end of file
Line	80	Continue search of file
Lines	90-110	Print result and stop
Lines	120-140	Print 'not found' message and stop

Note that the program contains two CLOSEIN statements: one if the country is found, and one if the file does not contain the country.

Read and write

The third and most useful type of file-handling program is a read and write or file management program. As the title suggests, the program assumes that a file already exists, which can then be

loaded into the computer, interrogated, changed and written back on to tape.

```
10 CLS
20 DIM A$(50),B$(50)
30 LET I=1
40 OPENIN "CAPITAL"
50 INPUT #9,A$(I)
60 INPUT #9,B$(I)
70 LET I=I+1
80 IF NOT EOF THEN GOTO 50
90 CLOSEIN
100 CLS
110 PRINT TAB(10)"MENU"
120 PRINT TAB(10)"===="
130 PRINT
140 PRINT "1....DISPLAY FILE"
150 PRINT "2....ADD NEW RECORDS"
160 PRINT "3....DELETE RECORD"
170 PRINT "4....CHANGE RECORD"
180 PRINT "5....STOP"
190 PRINT:PRINT
200 INPUT "SELECTION =";S
210 ON S GOTO 1000,2000,3000,4000,5000
220 GOTO 100
1000 REM ** DISPLAY FILE **
1010 CLS
1020 PRINT "COUNTRY","CAPITAL"
1030 PRINT "=======","======="
1040 FOR J=1 TO I-1
1050 PRINT A$(J),B$(J)
1060 NEXT J
1070 LOCATE 20,10:INPUT "PRESS ENTER TO
RETURN ";Z$
1080 GOTO 100
2000 REM ** NEW RECORDS **
2010 CLS
2020 INPUT "COUNTRY=";A$(I)
2030 INPUT "CAPITAL=";B$(I)
2040 LET I=I+1
2050 GOTO 100
3000 REM ** DELETE RECORD **
3010 INPUT "COUNTRY=";C$
3020 FOR J=1 TO I-1
3030 IF C$=A$(J) THEN GOTO 3080
3040 NEXT J
3050 PRINT "NO SUCH RECORD"
3060 LOCATE 20,10:INPUT "PRESS ENTER TO
RETURN ";Z$
```

```
3070 GOTO 100
3080 FOR K=J TO I-2
3090 LET A$(K)=A$(K+1)
3100 LET B$(K)=B$(K+1)
3110 NEXT K
3120 LET I=I-1
3130 GOTO 100
4000 REM ** CHANGE RECORD **
4010 CLS
4020 INPUT "COUNTRY=";C$
4030 FOR J=1 TO I-1
4040 IF C$=A$(J) THEN GOTO 4090
4050 NEXT J
4060 PRINT "NO SUCH RECORD"
4070 LOCATE 20,10:INPUT "PRESS ENTER TO
RETURN ";Z$
4080 GOTO 100
4090 INPUT "NEW COUNTRY=";A$(J)
4100 INPUT "NEW CAPITAL=";B$(J)
4110 GOTO 100
5000 REM ** SAVE FILE **
5010 OPENOUT "CAPITAL"
5020 FOR J=1 TO I-1
5030 PRINT #9,A$(J)
5040 PRINT #9,B$(J)
5050 NEXT J
5060 CLOSEOUT
5070 END
```

Comments

Lines	10-30	Initialise variables
Lines	40-90	Transfer data from file into RAM
Lines	100-220	Print main menu and make selection
Lines	1000-1080	Display all records on the screen
Lines	2000-2050	Input data for new record
Lines	3000-3130	Delete record from file
Lines	4000-4110	Change or edit an existing record
Lines	5000-5070	Transfer amended file on to tape

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 so the programmer has to resort to the use of files.

In Chapter 3 all the techniques considered here are used to create the 'educational database' which is used by most of the programs throughout this book.

3. The Database

Most of the programs contained within this book rely totally on a central educational database containing words and their meanings which represent information on a wide variety of subjects. The programs are designed to make full use of this information, and so the larger the database, the more efficient the individual programs become.

The database itself is designed to be expandable, and this chapter therefore contains two programs: the first initialises the database, while the second is an editor which provides the user with numerous facilities to check, extend or correct the information.

Because each program requires different information in a different format, the database is designed to be as flexible as possible, containing words and clues as well as words together with a classification group. The database is therefore divided into two sections: the first contains the actual information, the second the meanings of the various classifications.

Database Program 1

The first of the two database programs is used to initialise the file, putting a very limited amount of information on to tape together with the delimiter and the definition of the single category used.

The program should be entered and then run so that the file is saved on to tape. Because the database is so important it should be placed on both sides of a suitable cassette. As this program is only used for initialising the file, it will only be required once and need not be saved.

```
10 CLS
20 DIM A$(20),B$(20)
30 FOR I=1 TO 5
40 READ A$(I),B$(I)
50 NEXT I
60 PRINT TAB(10)"INITIALIZING FILE"
```

```
70 PRINT TAB(10)"============ ===="
80 PRINT:PRINT
90 PRINT "INPUT A NEW TAPE"
100 OPENOUT "EDBASE"
110 FOR I=1 TO 5
120 PRINT#9,A$(I)
130 PRINT#9,B$(I)
140 NEXT I
150 READ A$,B$
160 PRINT#9,"XXX"
170 PRINT#9,A$
180 PRINT#9,B$
190 CLOSEOUT
200 IF FLAG=1 THEN GOTO 240
210 PRINT "INPUT SECOND TAPE"
220 LET FLAG=1
230 GOTO 70
240 END
250 DATA LION,AN,TIGER,AN,ELEPHANT,AN,HIPPOPOTAMUS
,AN
260 DATA AN,ANIMAL
```

Comments

Lines	10-50	Initialise variables
Lines	60-90	Print information on the screen
Lines	100-190	Save data onto tape
Line	200	Check for number of copies of database. If FLAG=1 then end
Lines	210-230	If appropriate make second copy
Lines	240-260	Data for file

When the program has been executed there will be two copies of the EDBASE database stored on tape, with the structure shown below:

	FIELD 1	FIELD 2
RECORD 1	LION	AN
RECORD 2	TIGER	AN
RECORD 3	MONKEY	AN
RECORD 4	ELEPHANT	AN
RECORD 5	HIPPOPOTAMUS	AN
DELIMITER	XXX	
RECORD C1	AN	ANIMAL
	EOF	

The database as it now stands is of little use, but it is in a condition to be extended by the second program, shown below.

Database Program 2

The program listed here is a full DBMS (database management system), used to manipulate the EDBASE database.

```
10 CLS
15 LOCATE 14,3:PRINT"EDBASE EDITOR":LOCATE 14,4:PR
INT"====== ======":LOCATE 2,13:PRINT"PLACE EDBASE
CASSETTE INTO DATACORDER":PRINT"      PRESS PLAY A
ND THEN ANY KEY"
20 DIM A$(1100),B$(1100),C$(50),CL$(50)
30 LET I=1
40 OPENIN "!EDBASE"
50 INPUT#9,A$(I)
60 IF A$(I)="XXX" THEN GOTO 100
70 INPUT#9,B$(I)
80 LET I=I+1
90 GOTO 50
100 LET J=1
110 INPUT#9,CL$(J)
120 INPUT#9,C$(J)
130 IF EOF THEN GOTO 160
140 LET J=J+1
150 GOTO 110
160 CLOSEIN
170 CLS
180 PRINT TAB(18)"MENU"
190 PRINT TAB(18)"===="
200 PRINT
210 PRINT "        1....PRINT COMPLETE FILE"
220 PRINT "        2....CHANGE ENTRY"
230 PRINT "        3....EXTEND EDBASE"
240 PRINT "        4....SAVE EDBASE"
245 LOCATE 20,15:PRINT I-1;"WORDS IN EDBASE"
250 LOCATE 1,10:INPUT "SELECTION=";Z
260 IF Z>4 THEN GOTO 170
270 ON Z GOTO 1000,2000,3000,4000
1000 REM **PRINT FILE**
1010 CLS
1020 LOCATE 14,2:PRINT "DISPLAY FILE":PRINT:PRINT
1040 FOR K=1 TO I-1
1050 PRINT "WORD=";A$(K)
1060 PRINT "CLUE=";B$(K)
1070 NEXT K
```

```
1080 LOCATE 20,10:INPUT"ENTER TO CONTINUE ";Z$
1090 GOTO 170
2000 REM **CHANGE ENTRY**
2010 CLS
2020 LOCATE 14,2:PRINT "CHANGE ENTRY"
2030 LOCATE 20,15:PRINT I-1;"WORDS IN EDBASE"
2050 LOCATE 3,5:INPUT "WORD=";W$
2060 FOR K=1 TO I-1
2070 IF A$(K)=W$ THEN GOTO 2115
2080 NEXT K
2090 PRINT "NO SUCH ENTRY"
2100 GOTO 2140
2115 LOCATE 3,7:PRINT "CLUE=  ";B$(K)
2120 LOCATE 3,9:INPUT "NEW WORD=";A$(K)
2130 LOCATE 3,11:INPUT "NEW CLUE=";B$(K)
2140 LOCATE 10,23:LINE INPUT "PRESS ENTER TO CONT.
";Z$
2150 GOTO 170
3000 REM **EXTEND EDBASE**
3010 CLS:LOCATE 20,15:PRINT I-1;"WORDS IN EDBASE":
LOCATE 14,2:PRINT"ADD NEW WORDS"
3020 LOCATE 3,5:INPUT "WORD=";W$
3025 IF W$="END" THEN GOTO 170
3030 FOR K=1 TO I-1
3040 IF A$(K)=W$ THEN GOTO 3000
3045 NEXT K
3050 LOCATE 3,7:INPUT "CLUE=";B$(I)
3060 LET A$(I)=W$
3070 LET I=I+1
3080 GOTO 3000
4000 REM **SAVE EDBASE**
4010 CLS
4020 LOCATE 15,2:PRINT "SAVE EDBASE":PRINT:PRINT
4040 FOR K=1 TO I-1
4045 IF LEN(B$(K))<>2 THEN GOTO 4120
4050 FOR L=1 TO J
4060 IF CL$(L)=B$(K) THEN GOTO 4120
4070 NEXT L
4080 PRINT "CLASSIFICATION  =  ";B$(K)
4090 INPUT "CLASS DEFINITION=";C$(J)
4100 LET CL$(J)=B$(K)
4110 LET J=J+1
4120 NEXT K
4130 PRINT:PRINT:PRINT "    PLEASE WAIT BUFFERING
IN PROCESS"
4140 LOCATE 1,22:PRINT SPACE$(40):LOCATE 1,23:PRIN
T" PLACE EDBASE CASSETTE INTO DATACORDER  ";:PRINT
" PRESS PLAY & RECORD  AND THEN ANY KEY  ";:PRINT
SPACE$(40);
```

```
4150 OPENOUT "!EDBASE"
4160 FOR K=1 TO I-1
4170 PRINT#9,A$(K)
4180 PRINT#9,B$(K)
4190 NEXT K
4200 PRINT#9,"XXX"
4210 FOR K=1 TO J
4220 PRINT#9,CL$(K)
4230 PRINT#9,C$(K)
4240 NEXT K
4250 CLOSEOUT
4260 IF FLAG=1 THEN END
4270 PRINT "INPUT SECOND TAPE"
4280 LET FLAG=1
4290 GOTO 4130
```

Comments

Lines 10-30	Initialise variables
Lines 40-160	Input database, including information on classifications from tape
Lines 170-280	Display and make selection from main-menu
Lines 290-380	Print complete file on to screen, one record at a time
Lines 390-530	Edit section. User inputs a word with the computer locating the relevant record so that alterations can be made
Lines 540-640	Short section, enabling the user to increase the database
Lines 670-950	This is used to save the amended database on to tape. The various two-letter classifications are first checked. When this has been completed (line 780), the two copies are saved on to tape.

Because of the length of the database and the somewhat clumsy tape file-handling facilities of the Amstrad, the above program will take a long time to run, with the buffering taking several minutes each time an OPENIN, OPENOUT, CLOSEIN or CLOSEOUT statement is encountered.

EDBASE

There follow several pages of words and clues which should be placed into the database using the editor program above (Database Program 2). Although it is not *necessary* to enter all of these words, it is recommended that the majority are placed in the database so that the various programs will contain a lot of variety.

It can be seen that the words are divided into three categories for convenience, those with classifications, those with clues and finally those without clues (used for the card game in Chapter 9). When entering the final section it is necessary to place an asterisk (*) for the clue, indicating that these words have been included for the card game only.

WORDS WITH CLASSIFICATIONS
==========

LION	AN	TIGER	AN
ELEPHANT	AN	ZEBRA	AN
HORSE	AN	DONKEY	AN
GIRAFFE	AN	COW	AN
PIG	AN	MONKEY	AN
GORILLA	AN	LONDON	ET
YORK	ET	CARDIFF	ET
DOVER	ET	BRISTOL	ET
SHARK	FI	COD	FI
PLAICE	FI	SALMON	FI
PIKE	FI	CARP	FI
ROACH	FI	HADDOCK	FI
STINGRAY	FI	TENCH	FI
CAMEL	AN	DEER	AN
FOX	AN	WOLF	AN
RABBIT	AN	HARE	AN
ESSEX	EC	KENT	EC
SUFFOLK	EC	VAN	TR
MOUSE	AN	BADGER	AN
MOLE	AN	DICKENS	AU
ASIMOV	AU	MOOSE	AN
ANTEATER	AN	KANGAROO	AN
PANDA	AN	DEVON	EC
CORNWALL	EC	SOMERSET	EC
EXETER	ET	SHAKESPEARE	AU
GOAT	AN	LLAMA	AN
PUMA	AN	CHEETAH	AN

NORFOLK	EC	TRUMPET	MI
DRUM	MI	GUITAR	MI
ORGAN	MI	PIANO	MI
VIOLIN	MI	CELLO	MI
FLUTE	MI	BASSOON	MI
ROSE	FL	TULIP	FL
PANSY	FL	PRIMROSE	FL
BRONTE	AU	GREENE	AU
DAISY	FL	HARP	MI
HYENA	AN	ORWELL	AU
BISON	AN	ASS	AN
PENGUIN	BI	SEALION	AN
OXFORD	ET	WELLS	AU
BANJO	MI	CYMBAL	MI
CLARINET	MI	OBOE	MI
APPLE	FR	ORANGE	FR
BANANA	FR	PEACH	FR
GRAPE	FR	SHUTE	AU
JASON	BN	MELON	FR
APRICOT	FR	LEMON	FR
LIME	FR	BEAR	AN
NORWICH	ET	REINDEER	AN
OSTRICH	BI	TROUT	FI
RED	CO	GREEN	CO
YELLOW	CO	PINK	CO
MAGENTA	CO	WHITE	CO
CYAN	CO	BLACK	CO
LYNX	AN	SQUIRREL	AN
ANT	IN	FLEA	IN
BEETLE	IN	GIBBON	AN
BAGPIPES	MI	BEE	IN
HERRING	FI	BUFFALO	AN
MOTH	IN	EARWIG	IN
FRANCE	WC	GERMANY	WC
CHINA	WC	INDIA	WC
RUSSIA	WC	TIBET	WC
PAKISTAN	WC	JAPAN	WC
POLAND	WC	BULGARIA	WC
NORWAY	WC	FINLAND	WC
SWEDEN	WC	ICELAND	WC
EGYPT	WC	CHILE	WC
PERU	WC	BELGIUM	WC
SPAIN	WC	PORTUGAL	WC
ITALY	WC	GREECE	WC
ISRAEL	WC	YEMEN	WC
FOOTBALL	SP	CRICKET	SP
GOLF	SP	BASEBALL	SP
BOXING	SP	BOWLS	SP
AMERICA	WC	BARBARA	GN
AUSTRIA	WC	HOLLAND	WC

HOCKEY	SP	LACROSSE	SP
TENNIS	SP	SQUASH	SP
DARTS	SP	SNOOKER	SP
ASIA	CT	MACLEAN	AU
TAPIR	AN	SLOTH	AN
EUROPE	CT	AFRICA	CT
POPPY	FL	RUGBY	SP
SAILING	SP	SKIING	SP
FENCING	SP	PLUM	FR
HARDY	AU	KOREA	WC
LEON	BN	LIBYA	WC
ETHIOPIA	WC	YEN	CU
DOLLAR	CU	FRANC	CU
LIRA	CU	BRAHMS	GC
WALTON	GC	BACH	GC
ROSSINI	GC	BRITTEN	GC
MOZART	GC	POTATO	VE
CARROT	VE	TOMATO	FR
BEAN	VE	MARROW	VE
LETTUCE	VE	ONION	VE
CABBAGE	VE	SWEDE	VE
PARSNIP	VE	VIOLA	MI
NORMAN	BN	LISZT	GC
CHERRY	FR	LEEDS	ET
HAMSTER	AN	FROG	AN
TOAD	AN	BLUE	CO
BRAZIL	WC	COLUMBIA	WC
SHOOTING	SP	SKATING	SP
JUDO	SP	SWIMMING	SP
PERCH	FI	FLOUNDER	FI
STRAUSS	GC	WAGNER	GC
MAHLER	GC	SCHUBERT	GC
MARY	GN	HAYDN	GC
GREIG	GC	JONATHAN	BN
CARBON	EL	IRON	EL
GOLD	EL	SILVER	EL
HYDROGEN	EL	HELIUM	EL
ZINC	EL	TIN	EL
MERCURY	EL	OXYGEN	EL
TITANIUM	EL	COPPER	EL
MAGNESIUM	EL	SILICON	EL
KRYPTON	EL	BARIUM	EL
RHODIUM	EL	TUNGSTEN	EL
URANIUM	EL	NIGERIA	WC
UGANDA	WC	ZAMBIA	WC
MOON	AL	SUN	AL
STAR	AL	VENUS	AL
EARTH	AL	MARS	AL
JUPITER	AL	SATURN	AL
NEPTUNE	AL	URANUS	AL

PLUTO	AL	ASTEROID	AL
COMET	AL	JACK	BN
CLARKE	AU	METEOR	AL
CAP	CL	SHOE	CL
SOCK	CL	BOOT	CL
JACKET	CL	COAT	CL
SLIPPER	CL	SCARF	CL
TROUSERS	CL	SKIRT	CL
SHIRT	CL	JUMPER	CL
HAT	CL	SHORTS	CL
JEANS	CL	PANTS	CL
KNICKERS	CL	BRA	CL
VEST	CL	GLOVE	CL
MITTEN	CL	CARDIGAN	CL
CUP	KU	GLASS	KU
PLATE	KU	MUG	KU
KNIFE	KU	FORK	KU
SPOON	KU	DISH	KU
POT	KU	PAN	KU
MICROWAVE	KU	AARDVARK	AN
WHAM	PG	SWAN	BI
SAUCEPAN	KU	ARM	HA
LEG	HA	MOUTH	HA
NOSE	HA	HAND	HA
FOOT	HA	ANKLE	HA
TOE	HA	FINGER	HA
THUMB	HA	BACK	HA
KNEE	HA	BEATLES	PG
THROAT	HA	STOMACH	HA
CHEST	HA	BREAST	HA
BIN	KU	QUASAR	AL
PULSAR	AL	LACES	CL
TONGUE	HA	SPINE	HA
NERVES	HA	BRAIN	HA
TONSILS	HA	GUT	HA
HANDEL	GC	ROE	AN
XEON	EL	FEZ	CL
SPIT	KU	DUCK	BI
GOOSE	BI	CAR	TR
BUS	TR	PLANE	TR
JET	TR	TRAIN	TR
SHIP	TR	BALLOON	TR
BIKE	TR	TAXI	TR
CHURCH	BU	STATION	BU
FLAT	BU	HUT	BU
SHED	BU	SHOP	BU
LIBRARY	BU	HOUSE	BU
MUSEUM	BU	BUNGALOW	BU
GARAGE	BU	SCHOOL	BU
COLLEGE	BU	PUB	BU

HOTEL	BU	MANSION	BU
CASTLE	BU	KEEP	BU
CABIN	BU	CHALET	BU
LADLE	KU	MOSQUE	BU
CITADEL	BU	TEA	BE
COFFEE	BE	BEER	BE
LAGER	BE	CIDER	BE
ALE	BE	PORT	BE
WINE	BE	BRANDY	BE
WHISKY	BE	SHERRY	BE
GIN	BE	ADVOCAAT	BE
LEMONADE	BE	PLATYPUS	AN
KILT	CL	BELT	CL
BRACES	CL	GARTER	CL
TIGHTS	CL	PILCHARD	FI
TUNA	FI	PICCOLO	MI
ROOK	BI	GULL	BI
EAGLE	BI	FALCON	BI
ROBIN	BI	HAWK	BI
SPARROW	BI	PANTHER	AN
PELICAN	BI	KESTREL	BI
TEMPLE	BU	DOVE	BI
WREN	BI	TUBA	MI
TROMBONE	MI	OFFICE	BU
OAK	TE	ASH	TE
BEECH	TE	WILLOW	TE
MUSCLE	HA	MOSQUITO	IN
MARTIN	BI	TIT	BI
MANTIS	IN	MANDOLIN	MI
LYRE	MI	LUTE	MI
ABBA	PG	LIZARD	AN
OWL	BI	LARK	BI
OSPREY	BI	PUFFIN	BI
HORN	MI	URN	KU
ROBERT	BN	GNU	AN
EURHYTHMICS	PG	LEU	CU
DAVID	BN	STEPHEN	BN
SNAIL	AN	RICHARD	BN
GEORGE	BN	PETER	BN
JOHN	BN	SIMON	BN
PHILIP	BN	ROGER	BN
HENRY	BN	PATRICK	BN
JAMES	BN	NIGEL	BN
BRIAN	BN	PAUL	BN
ANDREW	BN	ALAN	BN
GRAHAM	BN	IAN	BN
SCOTT	BN	TIMOTHY	BN
BENJAMIN	BN	JAW	HA
JAY	BI	MAGPIE	BI
NICHOLAS	BN	LODGE	BU

NEIL	BN	VINCENT	BN
ANN	GN	ELIZABETH	GN
SUSAN	GN	JOYCE	GN
KATHERINE	GN	PAT	GN
PAMELA	GN	MARGARET	GN
LUCY	GN	SARAH	GN
HELEN	GN	RACHAEL	GN
REBECCA	GN	JANE	GN
SAMANTHA	GN	GINA	GN
MELISSA	GN	THERESA	GN
EMMA	GN	SALLY	GN
CLARE	GN	ELLA	GN
ALISON	GN	MANDY	GN
XENON	EL	JANET	GN
RUDD	FI	DIANA	GN
WILLIAM	BN	SQUARE	SH
TRIANGLE	SH	RECTANGLE	SH
PENTAGON	SH	ADAM	BN
ARMADILLO	AN	ARTHUR	BN
ASP	AN	BARN	BU
BASIL	BN	BREAM	FI
JANUARY	MO	FEBRUARY	MO
MARCH	MO	APRIL	MO
MAY	MO	JUNE	MO
JULY	MO	AUGUST	MO
SEPTEMBER	MO	OCTOBER	MO
NOVEMBER	MO	DECEMBER	MO
MONDAY	DW	TUESDAY	DW
WEDNESDAY	DW	THURSDAY	DW
FRIDAY	DW	SATURDAY	DW
SUNDAY	DW	XYLOPHONE	MI

CLASSIFICATIONS

AN	ANIMAL
ET	ENGLISH TOWN
FI	FISH
EC	ENGLISH COUNTY
TR	MODE OF TRANSPORT
MI	MUSICAL INSTRUMENT
FL	FLOWER
FR	FRUIT
CO	COLOUR
IN	INSECT
WC	COUNTRY
SP	SPORT
CT	CONTINENT
CU	CURRENCY

GC	GREAT COMPOSER
VE	VEGETABLE
CC	CAPITAL CITY
EL	ELEMENT
AL	ASTRONOMICAL BODY
CL	ITEM OF CLOTHING
KU	KITCHEN UTENSIL
HA	PART OF THE BODY
BU	BUILDING
BE	BEVERAGE
BI	BIRD
TE	TREE
AU	AUTHOR
PG	POP GROUP
BN	BOY'S NAME
GN	GIRL'S NAME
MO	MONTH
DW	DAY OF WEEK
SH	SHAPE

WORDS WITH CLUES

USE	APPLY
EGAD	BY GOD
OGLE	STARE
UGLY	HORRIBLE
CAT	FELINE
DOG	CANINE
PUZZLE	TEASER
FIGS	EXOTIC FRUITS
RAT	RODENT
AFTER	BEHIND
EARL	ARISTOCRAT
TRAVEL	JOURNEY
ABIDE	WAIT FOR
SQUID	SEA CREATURE
OCTOPUS	SEA CREATURE
TORCH	HAND LIGHT
YALE	AMERICAN UNIVERSITY
DOVETAIL	A JOINT
YEARN	PINE
NOD	BECK
FACT	TRUTH
FAME	GLORY
HOLINESS	BLESSEDNESS
HARM	DAMAGE
LOBSTER	SEA CREATURE
RAID	ATTACK

MUD	DIRT
GANG	BAND
GASP	PANT
CONFIRM	ASSURE
PARIS	CAPITAL OF FRANCE
BERLIN	CAPITAL OF EAST GERMANY
CONFOUND	AMAZE
BONN	CAPITAL OF WEST GERMANY
LISBON	CAPITAL OF PORTUGAL
MADRID	CAPITAL OF SPAIN
ATHENS	CAPITAL OF GREECE
ROME	CAPITAL OF ITALY
BRUSSELS	CAPITAL OF BELGIUM
TRIPOLI	CAPITAL OF LIBYA
TOKYO	CAPITAL OF JAPAN
MOSCOW	CAPITAL OF RUSSIA
PEKING	CAPITAL OF CHINA
CANBERRA	CAPITAL OF AUSTRALIA
BET	STAKE
RELY	DEPEND
POVERTY	PAUPERISM
WARSAW	CAPITAL OF POLAND
VIENNA	CAPITAL OF AUSTRIA
MENTION	REFER TO
LID	PAN TOP
TOP	SPINNING ---
COOKER	STOVE
STOVE	COOKER
FRIDGE	COOLER
FREEZER	ICE BOX
SINK	WASHING BOWL
DRYER	TUMBLE -----
SUMMER	SEASON
EYE	HUMAN LIGHT RECEPTOR
EAR	HUMAN SOUND RECEPTOR
HAIR	FOUND ON HUMAN HEAD
SHIN	LEGBONE
LAB	SCIENCE WORKSHOP
SCISSORS	CUTTING IMPLEMENT
JADE	SEMI PRECIOUS STONE
BRUSH	----- AND COMB
EXTINCT	OBSOLETE
KIDNEY	A VITAL ORGAN
HEART	A VITAL ORGAN
LIVER	A VITAL ORGAN
LUNGS	VITAL ORGANS
ARTERIES	BLOOD CANALS
BAT	FLYING RODENT
LOUD	NOISY
ROUND	CIRCULAR

MERCHANT	TRADER
FRACTION	PORTION OF
ULNA	ARM BONE
POINT	DIRECT
POLISH	SHINE
PLUMP	FAT
JIVE	DANCE
ICE	FROZEN WATER
SCAR	MARK
EXOTIC	STRANGE
EXIT	WAY OUT
INK	WRITING FLUID
EASE	COMFORT
ALSO	AS WELL
ACID	NOT ALKALINE
UNIT	ONE
WET	NOT DRY
WAR	BATTLE
EJECT	THROW OUT
FORD	CROSSING
KEY	OPENER
QUIET	CALM
SABRE	SWORD
NAKED	BARE
PINT	FLUID MEASURE
GALLON	FLUID MEASURE
SMART	TIDY
SMALL	TINY
IDOL	DEITY
HICCUP	NOT TO PLAN
GOYA	ARTIST
COOK	PREPARE FOOD
DESPISE	SCORN
DESPAIR	LOSE HOPE
CONFLICT	BATTLE
SAY	DECLARE
APPROACH	ADVANCE
ACRE	LAND MEASURE
GAP	BREACH
COO	SOUND MADE BY DOVE
KIN	RELATION
VICINITY	NEARNESS
VERTICAL	UPRIGHT
DOLE	ALLOCATE
CLOWN	BUFFOON
ABUNDANT	AMPLE
SUM	TOTAL
HUMID	MOIST
STRIP	UNDRESS
WOO	COURT

UNEASY	DISTURBED
VERSE	POETRY
SIXTY	FEMALE RETIREMENT AGE
KING	MONARCH
KNIT	BIND
KIND	GENUS
ANNALS	ARCHIVES
ANSWER	SOLUTION
SOPPING	SOAKED
ASIDE	APART
INAPT	UNFIT
ADAPT	EVOLVE
ORBIT	COURSE
BEMUSE	STUPEFY
ALMOND	A NUT
INNER	INTERIOR
SADLY	SORROWFULLY
QUEEN	MONARCH
PLASTIC	MATERIAL
SUGAR	SWEET
IGNORE	NEGLECT
PUNGENT	SMELLING NASTY
FEE	BILL
FELON	CONVICT
HORDE	MULTITUDE
CUSTOM	CONVENTION
DAINTY	CHARMING
ACRID	PUNGENT
MOTTO	INSCRIPTION
SHARP	POINTED
START	COMMENCE
ZOO	ANIMAL PARK
ZIP	FASTENER
PHOENIX	LEDGENDARY BIRD
ORC	MYTHICAL CREATURE
ADVISE	RECOMMEND
MINGLE	BLEND
OFFAL	LIVER
OFFER	TENDER
VASE	CONTAINER FOR FLOWERS
JESUS	SAVIOUR
SINGE	LIGHTLY BURN
SIREN	AIR-RAID WARNING
SISSY	EFFEMINATE BOY
SCREW	TIGHTEN
DRAGON	ST.GEORGE´S ADVERSARY
JEWEL	PRECIOUS STONE
KERRY	COUNTY IN IRELAND
KHAKI	ARMY FABRIC
LOGIC	CHAIN OF REASONING

MAORI	NATIVE OF NEW ZEALAND
SIGHT	FACULTY OF VISION
WEDGE	TAPERED PIECE OF WOOD
VALUE	WORTH
ALLAH	MOSLEM GOD
ALLOY	A MIXTURE OF METALS
PARADOX	MYSTERY
SLY	ARTFUL
UNABLE	INCAPABLE
VIE	CONTEST
IMMORAL	DEBAUCHED
BESTOW	IMPART
CHOP	MINCE
PORTION	ONE HELPING
MYTH	LEGEND
NAG	HARASS
NEUTRAL	IMPARTIAL
REDRAW	DRAW AGAIN
ORAL	VOCAL
PEER	ASSOCIATE
POWERFUL	FORCEFUL
SNEER	DERIDE
SUE	PETITION
SURGE	SWELL
TAMPER	ALTER
TARNISH	DISCOLOUR
TETHER	MANACLE
THICKET	WOODLAND
THRIFT	ECONOMY
FAST	GO WITHOUT FOOD
TROUBLE	INCONVENIENCE
UNCOUTH	BOORISH
VENT	EXPRESS
EVIL	CORRUPT
EXCITE	AROUSE
FACTION	PARTY
REQUEST	ASK
DIVINE	FATHOM
FLEET	NAVY
FLOW	GLIDE
FREE	LIBERATE
FRET	WORRY
FRIEND	COMPANION
FUSE	MERGE
FUN	MIRTH
GAGE	PLEDGE
SIDES	EDGES
CHATTER	TALK QUICKLY
PADRE	PRIEST
REFUSAL	REJECTION

GHASTLY	HORRIBLE
JETTY	PIER
JACOB	SON OF ISAAC
JUMBO	ELEPHANT
JUMPY	NERVOUS
FIREMAN	FIGHTS FIRES
POSTMAN	DELIVERS LETTERS
BADGE	EMBLEM
SATIN	FINE CLOTH
BIPED	TWO-LEGGED
ERROR	MISTAKE
ASSAY	EXAMINE
APPLAUD	CONGRATULATE
CHEAP	OF LITTLE VALUE
HARDY	STOUT-HEARTED
SKILL	ABILITY
RAVINE	GORGE
LIP	PART OF MOUTH
PITY	COMPASSION
CRY	LAMENT
PINE	YEARN
GAMBOL	FROLIC
EXTEND	LENGTHEN
PROSPER	FLOURISH
PRY	SEARCH
STEAL	PILFER
BASH	STRIKE HARD
SLEDGE	SNOWMOBILE
ELITE	TOP RANK
QUIT	ABANDON
QUIVER	ARROW HOLDER
FORK	KNIFE AND ----
COMPOSER	WRITER OF MUSIC
MORNING	TIME OF DAY
AFTERNOON	TIME OF DAY
NIGHT	DAY AND -----
AGAINST	IN OPPOSITON TO
AGINCOURT	FAMOUS BATTLE
HIT	STRIKE
AMAZE	WONDER
ANCHOR	FIX FIRMLY
ANCIENT	VERY OLD
ANGEL	CHERUB
ANNOY	IRRITATE
ANTIQUE	OLD AND VALUABLE
APPEAR	BECOME VISIBLE
ARCH	CURVED STRUCTURE
ARMOUR	DEFENSIVE CLOTHING
ARRANGE	PUT IN ORDER
ASK	REQUEST

AXE	CHOPPER
BALANCE	EQUILIBRIUM
BAMBOO	HOLLOW STEM
BARK	CRY OF DOG
BASIN	CIRCULAR VESSEL
BED	FLAT BASE
BEGIN	START
BENCH	LONG WOODEN SEAT
BERRY	SMALL FRUIT
BINARY	BASE TWO
BIOLOGY	SCIENCE OF LIFE
BIRTH	ORIGIN
BLANK	WITHOUT DETAIL
BLAME	FIND FAULT
BLAZE	BURN BRIGHTLY
COPY	IMITATION
BLUNDER	MISTAKE
BOSS	MASTER
BOTTLE	GLASS VESSEL
BRAKE	STOP
BRAVE	COURAGEOUS
BREEZE	GENTLE WIND
GALE	STRONG WIND
BUCKET	PAIL
CAIRO	CAPITAL OF EGYPT
CAFE	TEA SHOP
CANOE	SMALL BOAT
CAPE	CLOAK
LIQUEUR	POTENT DRINK
SYMBOL	SIGN
DETECT	FIND
INSTRUCTION	DIRECTION
FORCE	POWER
FRY	COOK IN FAT
BAKE	COOK IN OVEN
OAF	LOUT
QUICK	FAST
QUESTION	INTERROGATE
POSTER	PLACARD
REMOTE	FAR APART
CAUSE	REASON
CALL	SHOUT
OBSERVE	SIGHT
SPINSTER	UNMARRIED WOMAN
SOUND	NOISE
SPEAK	MAKE UTTERANCE
ROUTINE	NORMAL ACTION
NOOSE	HANGMAN'S LOOP
RACE	CHASE
RAISE	ELEVATE

TIE	SHIRT AND ---
RARITY	SCARCITY
LAD	YOUNG MAN
SALTY	SALINE
SECTS	FACTIONS
ELBOW	ARM JOINT
HABIT	TENDENCY
SUITE	ESCORT
PEPPER	SALT AND ------
ROOMY	EXTENSIVE
STRAP	BELT
ABBEY	PRIORY
SAKER	LARGE FALCON
LAIR	HOME OF BEAST
SALARY	INCOME
TOTAL	WHOLE
TOY	PLAYTHING
TRACE	OUTLINE
DAY	NIGHT AND ---
ULCER	BOIL
UMPIRE	JUDGE
BONE	DOG AND ----
VALLEY	DALE
VAST	BOUNDLESS
VET	ANIMAL DOCTOR
WARN	CAUTION
WARRANT	DECLARE
WARP	BEND
WILY	CUNNING
WINNOW	SEPARATE
WANT	REQUIRE
DOBERMANN	GUARD DOG
TOE	HEAD TO ---
PASS	NARROW PASSAGE
ACME	PINNACLE
BORDER	DIVIDING LINE
ABYSS	BOTTOMLESS PIT
BELLOW	SHOUT
WEEP	CRY
BRIM	EDGE
BULWARK	GUARD
CHAGRIN	MORTIFICATION
CEDE	ABDICATE
CAUSTIC	CORROSIVE
CLONE	EXACT COPY
CLOTHING	APPAREL
COMA	UNCONSCIOUS STATE
COERCE	COMPEL
DESPOT	AUTOCRAT
DICTATOR	DESPOT

DIFFUSE	SPREAD
ABOVE	HIGHER UP
BELOW	LOWER DOWN
ABSENT	NOT PRESENT
ABSOLUTE	COMPLETE
ABSORB	TAKE IN
ACCIDENT	MISFORTUNE
HOT	NOT COLD
COLD	NOT HOT
ACHE	PAIN
ACROBAT	GYMNAST
ACROSS	OTHER SIDE
ACT	DEED
ACTOR	DRAMATIC PERFORMER
POET	WRITER OF VERSE
SUPPER	EVENING MEAL
POKE	PUSH
DAILY	EVERY DAY
ANNUAL	ONCE A YEAR
MANUAL	PERFORMED BY HAND
HUNT	LOOK FOR
GUARD	PROTECT
BASIC	SIMPLE
JOB	TASK
NAP	SHORT SLEEP
CRIBBAGE	CARD GAME
TYPE	KIND
ENTER	GO IN
DELETE	ERASE
HARK	LISTEN
RETURN	GO BACK
ESCAPE	RUN AWAY
ARCTIC	POLAR REGION
ARROW	BOW AND -----
REMAIN	STAY
CHOOSE	SELECT
ATHLETE	RUNNER
ATLANTIS	LOST CITY
ATTEND	BE PRESENT
AUCTION	PUBLIC SALE
BABY	YOUNG CHILD
BACHELOR	UNMARRIED MAN
DRAW	SKETCH
TRANSPARENT	CLEAR
BAKER	MAKER OF BREAD
BALD	WITHOUT HAIR
CHOIR	SINGING GROUP
BANDIT	OUTLAW
BANNER	FLAG
BARE	UNCLOTHED

BARLEY	CEREAL
WHEAT	CEREAL
OAT	CEREAL
SAVE	RESCUE
ROAR	BELLOW
PAUSE	WAIT
LORD	PEER
BLINK	MOVE EYELID
TRAP	CAPTURE
BOAST	PRAISE ONESELF
TINSEL	CHRISTMAS DECORATION
SANDAL	FOOTWEAR
CLOG	WOODEN SHOE
NEED	WANT
NEWS	TIDINGS
ATLAS	BOOK OF MAPS
RELAX	REST
SPRING	SEASON
AUTUMN	SEASON
WINTER	SEASON

SINGLE WORDS

IN	US	THE	AN
A	TO	TOO	WE
IF	GO	UP	SIT
AT	ON	THEM	THAT
THERE	THEIR	HIS	HERS
HIM	HER	HE	SHE
THOSE	ALL	DROP	ABOUT
I	AND	WITH	ME
MY	OR	BY	AM

4. Type Attack

The greatest problem most people face when they first use a computer is locating the keys on a 'QWERTY' keyboard. This short program provides a colourful game which will help you become familiar with the keyboard. Although we cannot promise to convert you instantly into an audio touch typist, we are sure that you will gain valuable expertise from your time spent playing Type Attack.

Instructions

When the program is executed you are shown a series of coloured bands with numbers written beside them, corresponding to the number of points which will be awarded throughout the game. However, first you must input the level of play: you have a choice of 1 to 5, of which 1 is the easiest and 5 the most difficult. You are then requested to press the space bar to begin the game.

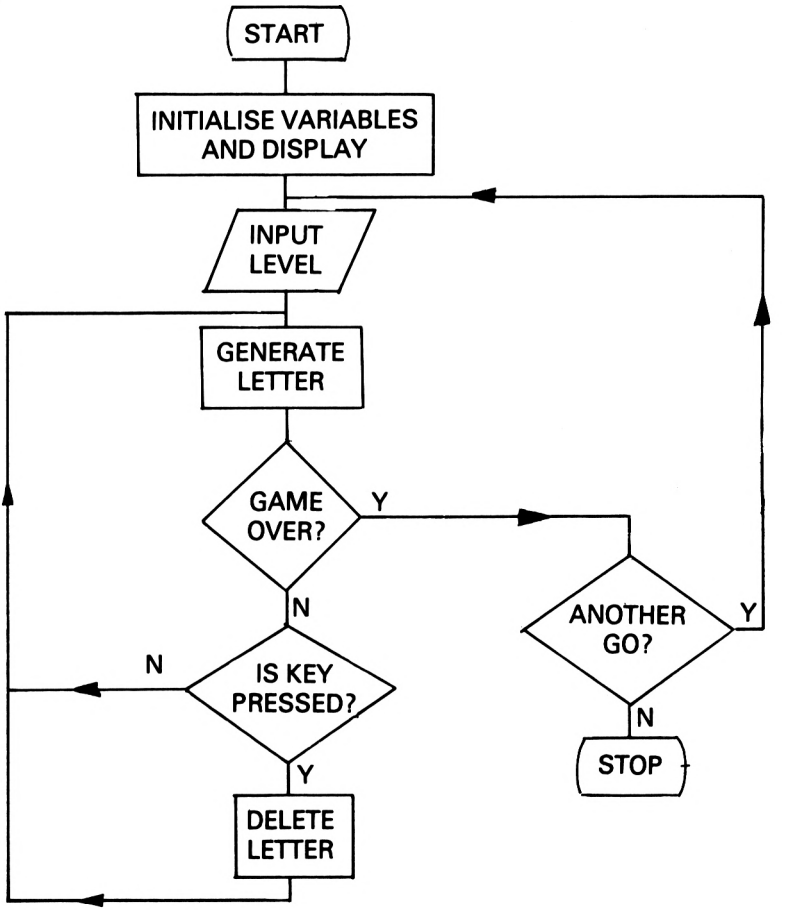
A series of letters appear descending down the screen. To destroy them, you must type the corresponding keys. Depending on how quickly you respond, you are awarded points which decrease the further the letter is down the screen. You are killed when one of the letters reaches the base line and the game ends.

If your score is greater than the high score then this is updated and you are asked if you wish to play again.

The program

Flowchart

The flowchart 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.

HI	High Score
A$()	The letters
D()	Distance down the screen
LEV$	The character representing the level
LEV	The numerical equivalent of LEV$
Y	Position within the window
W	Window containing letter
SCORE	The current score
Z$	Variable used for another go

Comments on the program

This is a very short program and consequently contains few subroutines.

Lines 10-220
This section defines the inks and windows and dimensions the arrays to be used during the game.

Lines 230-400
This is where the screen is constructed, showing the different bands of colour, the scores and the title.

Lines 410-460
The level is specified and on pressing the space bar the game commences.

Lines 470-530
This is where the random letters and columns for them to descend in are chosen. Line 470 limits the number to be produced on the screen when playing on a lower level.

Lines 540-680
Here the letters are moved and displayed in their new positions.

Lines 690-860
This section of code checks to see if the key of one of the letters has been pressed and if it has then the score is amended and the letter is removed.

Lines 870-1100
The end routine which amends the high score if necessary and asks whether the player wants another game.

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 INK 0,26
20 LET HI=100
30 CLS
40 MODE 0
50 DIM A$(18),D(18)
60 INK 0,26
70 INK 1,6
80 INK 2,2
90 INK 3,24
100 INK 4,20
110 INK 5,15
120 INK 6,8
130 INK 7,18
140 INK 8,0
150 INK 9,0
160 WINDOW#1,2,19,1,2
170 WINDOW#2,2,19,3,4
180 WINDOW#3,2,19,5,6
190 WINDOW#4,2,19,7,8
200 WINDOW#5,2,19,9,10
210 WINDOW#6,2,19,11,12
220 WINDOW#7,2,19,13,14
230 FOR I=1 TO 7
240 PAPER#I,I
250 PEN #I,8
260 CLS#I
270 NEXT I
280 PEN 8
290 FOR I=9 TO 2 STEP -1
300 LOCATE 1,20-(2*I):PRINT USING"#";I
```

```
310 LOCATE 20,20-(2*I):PRINT USING"#";I
320 NEXT I
330 PEN 9:LOCATE 1,18:PRINT STRING$(20,CHR$(143)):
PEN 8
340 PLOT 32,399,8:DRAWR 0,-270
350 PLOT 607,399,8:DRAWR 0,-270
360 LOCATE 5,20:PRINT "TYPE  ATTACK"
370 PEN 1
380 LOCATE 5,22:PRINT "SCORE    0"
390 PEN 2
400 LOCATE 2,24:PRINT "HI-SCORE ";HI
410 LOCATE 5,16:PRINT"INPUT LEVEL"
420 LET LEV$=INKEY$:IF LEV$<"1" OR LEV$>"5" THEN G
OTO 420
430 LET LEV=VAL(LEV$)
440 LOCATE 3,16:PRINT "PRESS SPACE BAR"
450 IF INKEY$<>" " THEN GOTO 450
460 LOCATE 3,16:PRINT SPACE$(16);
470 IF RND(1)>0.5+(LEV/10) THEN GOTO 550
480 LET R=2+INT(RND(1)*16)
490 IF A$(R)<>"" THEN GOTO 550
500 LET A$(R)=CHR$(65+INT(RND(1)*26))
510 LET C=C+1
520 SOUND 2,1000,10
530 LET D(R)=1
540 PEN 8
550 FOR I=2 TO 17
560 LET X=I
570 IF A$(I)="" THEN GOTO 680
580 LET W=INT((D(I)+1)/2)
590 IF D(I)/2=INT(D(I)/2) THEN LET Y=2 ELSE Y=1
600 IF W>7 THEN LET W=0:LET Y=D(I):LET X=I+1
610 LOCATE #W,X,Y:PRINT #W," "
620 LET D(I)=D(I)+INT((SCORE/(200-20*LEV))+1)
630 LET W=INT((D(I)+1)/2)
640 IF D(I)/2=INT(D(I)/2) THEN LET Y=2 ELSE Y=1
650 IF W>7 THEN LET W=0:LET Y=D(I):LET X=I+1
660 IF Y>17 THEN GOTO 870
670 LOCATE #W,X,Y:PRINT #W,A$(I)
680 NEXT I
690 LET Z$=INKEY$
700 IF Z$="" THEN GOTO 470
710 FOR I=2 TO 17
720 IF A$(I)=Z$ THEN GOTO 750
730 NEXT I
740 GOTO 470
750 LET A$(I)=""
760 LET C=C-1
770 LET X=I
780 LET W=INT((D(I)+1)/2)
```

```
790 IF D(I)/2=INT(D(I)/2) THEN LET Y=2 ELSE Y=1
800 IF W>7 THEN LET W=0:LET Y=D(I):LET X=I+1
810 LOCATE #W,X,Y:PRINT #W," "
820 IF W=0 THEN LET SCORE=SCORE+2 ELSE LET SCORE=S
CORE+(10-W)
830 PEN 1:LOCATE 12,22:PRINT USING "###";SCORE:PEN
 8
840 SOUND 1,80,10
850 GOTO 470
860 GOTO 860
870 REM ** END ROUTINE **
880 FOR I=80 TO 10 STEP -5:SOUND 7,I,10:NEXT I
890 FOR J=1 TO 3:FOR K=1 TO 8:INK 9,K:FOR L=1 TO 5
0:NEXT L,K,J
900 INK 9,0
910 FOR I=1 TO 10:LET Z$=INKEY$:NEXT I
920 IF SCORE>HI THEN LET HI=SCORE
930 PEN 2:LOCATE 2,24:PRINT "HI-SCORE ";HI:PEN 8
940 LET SCORE=0
950 FOR I=15 TO 17
960 LOCATE 3,I:PRINT SPACE$(16);
970 NEXT I
980 LOCATE 5,16:PRINT "ANOTHER GO ?"
990 LET Z$=INKEY$
1000 IF Z$="" THEN GOTO 990
1010 IF Z$="N" THEN END
1020 PEN 1:LOCATE 12,22:PRINT "  0":PEN 8
1030 FOR I=1 TO 7
1040 CLS#I
1050 NEXT I
1060 FOR I=2 TO 17
1070 LET A$(I)=""
1080 IF I>14 THEN LOCATE 3,I:PRINT SPACE$(16)
1090 NEXT I
1100 GOTO 280
```

5. Crossword Puzzler

One of the most effective ways of learning new words and their meanings is to do crossword puzzles. This program will provide the user with an endless supply of puzzles.

Instructions

When the program is loaded from tape and executed, a brief résumé of the playing instructions will appear on the screen and the database EDBASE will be transferred into the computer.

When the loading process is complete, there will be a substantial delay while the Amstrad organises its memory allocation. When this has been done you will be asked whether or not you require a time limit game, and 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 word positions 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.

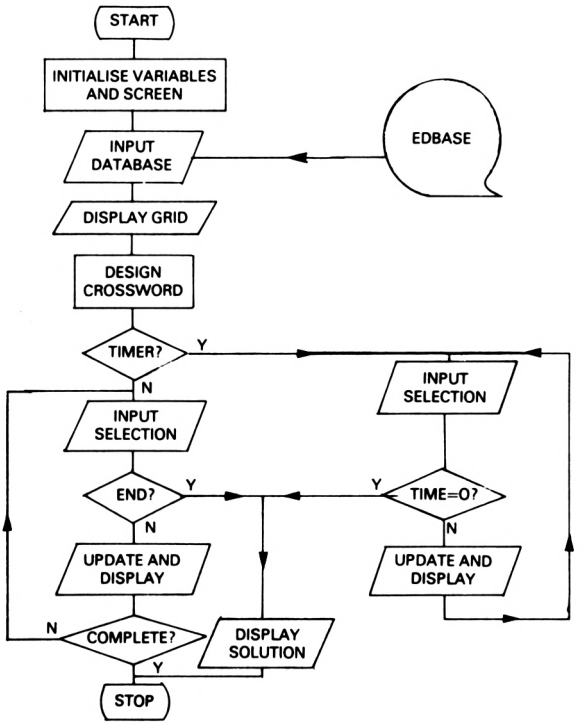
2. During the crossword design stage the message 'DESIGNING CROSSWORD' may flash, indicating that the program was unable to complete the puzzle due to the limitations of the database and a re-start is taking place.

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-five of the individual words are correct.

The program

Flowchart

The flowchart 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

The program is structured, and the following notes show the operation of each of the individual routines.

Lines 10-20
This section dimensions arrays and calls a subroutine to display instructions on the screen.

Lines 30-160
Opens a cassette file and transfers the information from the EDBASE 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 problem 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 10 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 '0', indicating a white square into which a letter can be placed, or a '*' indicating a black square.

Lines 310-860
The main section used to create the graphical representation on the screen.

Lines 870-1020
The control section for the design of the puzzle. 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
Initialises the timer if it is required.

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, legality checks, etc.

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

```
A$(1)=DBA3
```

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

Lines 1790-2050
This routine is used in the design stage to locate a word and clue to fit in a horizontal position on the grid.

Lines 2060-2270
This routine prints the current grid condition on the screen. It is used throughout the program to display the intermediate positions and at the end to display the final solution.

Lines 2280-2530
Similar to the routine at 1790, this is used for the vertical positions on the grid.

Lines 2540-2630
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 2640-3040
This routine controls the movement of the pointer and allows the user to input the answer. The answer is then checked for type ('END', '-' or a word, a legality check is performed and the appropriate action is taken.

Lines 3050-3070
A short routine to print error messages when an entry fails to pass a legality check.

Lines 3080-3250
This routine checks for a correct solution to the crossword by comparing the contents of the array C$() with COPY$().

Lines 3260-3620
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 remove, as it forms part of the word ASH.

```
    A
HOUSE
    H
```

Lines 3630-3810
This is a short timer routine.

Lines 3820-3930
A routine called very early in the program which prints a very brief résumé of the instructions 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 DIM A$(27),CS$(40,2),C$(14,14),COPY$(14,14),W$(
1000),CL$(1000),CN$(27),AN$(27)
20 GOSUB 3820
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
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
```

```
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,"30:00"
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$="D" THEN LET S=ST:LET F=WORDS:GOSUB 22
80
980 IF DIR$="D" AND TES=0 THEN LET S=1:LET F=ST:GO
SUB 2280
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=30: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 3630: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$(I),1);")":LET DO=DO+1
1210 NEXT I
1220 IF GAME=1 THEN LET GAME=0:RETURN
1230 GOSUB 2640
1240 IF GOVER=1 THEN GOTO 3690
1250 IF GUESS$="END" AND LIMIT=0 THEN GOTO 3700
1260 IF GUESS$="-" THEN GOTO 3260
1270 IF LEN(GUESS$)<>LEN(AN$(PE))THEN GOTO 3050
1280 LET X$=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 3050
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$,K,1)
1430 LET K=K+1
1440 NEXT J
1450 GOSUB 2060
1460 GOTO 3080
1470 FOR N=Y TO Y+L-1
1480 IF C$(X,N)="0" THEN GOTO 1510
1490 IF C$(X,N)=MID$(GUESS$,K,1) THEN GOTO 1510
1500 GOTO 3050
1510 LET K=K+1:NEXT N
1520 LET K=1
1530 FOR J=Y TO Y+L-1
1540 LET C$(X,J)=MID$(GUESS$,K,1)
1550 LET K=K+1
1560 NEXT J
1570 GOSUB 2060
1580 GOTO 3080
1590 GOTO 1590
1600 END
1610 RESTORE 1650
1620 FOR I=1 TO 25
1630 READ A$(I)
```

```
1640 NEXT I
1650 DATA AAA8,DBA4
1660 DATA AAC5,DEC4
1670 DATA AEE3,DHA4
1680 DATA AGC5,DJA4
1690 DATA AJA4,DMA4
1700 DATA DGE5,AGF5
1710 DATA DKF5,AJJ4
1720 DATA DMF8,ABH6
1730 DATA DDH3,AAJ6
1740 DATA DAI5,DCJ4
1750 DATA DEJ4,AEM5,DGK3
1760 DATA DIK3
1770 DATA AIL3
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$(M),K,1) THEN GOTO 1920
1910 GOTO 1960
1920 LET K=K+1:NEXT N
1930 GOSUB 2540
1940 IF OK=0 THEN GOTO 1960
1950 GOTO 1980
1960 NEXT M
1970 IF S<>1 THEN GOTO 2050 ELSE LET FAIL=1:GOTO 2
050
1980 LET K=1
1990 FOR J=X TO X+(L-1)
2000 LET C$(J,Y)=MID$(W$(M),K,1)
2010 LET K=K+1
2020 NEXT J
2030 LET TES=1
2040 GOSUB 2060
2050 RETURN
2060 REM ***TO BE REMOVED***
2070 PRINT CHR$(22)+CHR$(1);
2080 PAPER 0
2090 LET J=0
2100 FOR Y=370 TO 150 STEP -18
2110 LET I=0
2120 LET J=J+1
```

```
2130 FOR X=350 TO 570 STEP 18
2140 LET I=I+1
2150 PLOT 0,0,1
2160 MOVE X,Y
2170 TAG
2180 IF C$(I,J)="*" THEN PRINT CHR$(143);:GOTO 222
0
2190 IF C$(I,J)="0" THEN PRINT CHR$(32);: GOTO 222
0
2200 MOVE X,Y
2210 PRINT C$(I,J);
2220 TAGOFF
2230 NEXT X
2240 NEXT Y
2250 PRINT CHR$(22)+CHR$(0);
2260 CLS#2
2270 RETURN
2280 REM DESIGN VERTICALS
2290 LET X$=MID$(A$(P),2,1)
2300 LET Y$=MID$(A$(P),3,1)
2310 LET X=ASC(X$)-64
2320 LET Y=ASC(Y$)-64
2330 LET L=VAL(RIGHT$(A$(P),1))
2340 FOR M=S TO F
2350 IF LEN(W$(M))<>L THEN GOTO 2450
2360 LET K=1
2370 FOR N=Y TO Y+L-1
2380 IF C$(X,N)="0" THEN GOTO 2410
2390 IF C$(X,N)=MID$(W$(M),K,1) THEN GOTO 2410
2400 GOTO 2450
2410 LET K=K+1:NEXT N
2420 GOSUB 2540
2430 IF OK=0 THEN GOTO 2450
2440 GOTO 2470
2450 NEXT M
2460 IF S<>1 THEN GOTO 2530 ELSE LET FAIL=1:GOTO 2
530
2470 LET K=1
2480 FOR J=Y TO Y+(L-1)
2490 LET C$(X,J)=MID$(W$(M),K,1)
2500 LET K=K+1
2510 NEXT J
2520 LET TES=1
2530 RETURN
2540 REM **CHECK FOR REPETITION**
2550 LET OK=1
2560 FOR JK=1 TO NW-1
2570 IF AN$(JK)=W$(M) THEN LET OK=0:RETURN
2580 NEXT JK
2590 LET OK=1
```

```
2600 LET AN$(NW)=W$(M)
2610 LET CN$(NW)=CL$(M)
2620 LET NW=NW+1
2630 RETURN
2640 REM **CURSOR**
2650 CLS#2
2660 LOCATE#2,6,2
2670 PRINT#2,"MOVE CURSOR"
2680 PAPER 2
2690 LOCATE CX,CY
2700 FOR DEL=1 TO 100:NEXT
2710 IF INKEY(2)=0 THEN CY=CY+1:PRINT" "
2720 IF CY=19 AND CX=1 THEN CY=3:CX=9
2730 IF CY=14 AND CX=9 THEN CY=3:CX=1
2740 IF INKEY(0)=0 THEN CY=CY-1:PRINT" "
2750 IF CY=2 AND CX=1 THEN CY=13:CX=9
2760 IF CY=2 AND CX=9 THEN CY=18:CX=1
2770 IF INKEY(58)=0 THEN GOTO 2820
2780 LOCATE CX,CY
2790 PRINT CHR$(243)
2800 IF GOVER=1 THEN GOTO 3690
2810 GOTO 2690
2820 CLS#2:FOR LM=1 TO 20
2830 LET Z$=INKEY$
2840 NEXT LM
2850 LET PE=CY-2
2860 IF CX=1 THEN GOTO 2900
2870 FOR DE=1 TO 27
2880 IF LEFT$(A$(DE),1)="A" THEN LET PE=PE-1:IF PE
=0 THEN LET PE=DE:GOTO 2930
2890 NEXT DE
2900 FOR DE=1 TO 27
2910 IF LEFT$(A$(DE),1)="D" THEN LET PE=PE-1:IF PE
=0 THEN LET PE=DE:GOTO 2930
2920 NEXT DE
2930 PEN#2,1:LOCATE#2,1,1:PRINT#2,"THE CLUE IS:-"
2940 PEN#2,3
2950 IF LEN(CN$(PE))>2 THEN LOCATE#2,INT((23-LEN(C
N$(PE)))/2),2:PRINT#2,CN$(PE):GOTO 3000
2960 FOR JH=1 TO CLUES
2970 IF CS$(JH,1)=CN$(PE) THEN GOTO 2990
2980 NEXT JH
2990 LOCATE#2,INT((23-LEN(CS$(JH,2)))/2),2:PRINT#2
,CS$(JH,2)
3000 PEN#2,1
3010 LOCATE#2,1,3:PRINT#2,"ENTER WORD (- TO DELETE
)"
3020 PEN#2,3
3030 LOCATE#2,1,4:INPUT#2,"",GUESS$
3040 RETURN
```

```
3050 REM **PRINT ERROR MESSAGE**
3060 CLS#2
3070 GOTO 1080
3080 REM **CHECK FOR SOLUTION **
3090 FOR I=1 TO 13
3100 FOR J=1 TO 13
3110 IF C$(I,J)<>COPY$(I,J) THEN GOTO 1080
3120 NEXT J
3130 NEXT I
3140 LET D=REMAIN(1)
3150 FOR J=1 TO 10
3160 FOR I=1 TO 100
3170 INK 1,INT(RND*25)+1
3180 SOUND 1,I*10,1:SOUND 2,I/5,1
3190 IF SQ(1)<>4 THEN GOTO 3190
3200 NEXT I
3210 NEXT J
3220 INK 1,0
3230 CLS#2
3240 PEN#2,1
3250 LOCATE#2,4,3:PRINT#2,"PUZZLE COMPLETE":GOTO 3
770
3260 REM **REMOVE WORD**
3270 X$=MID$(A$(PE),2,1)
3280 Y$=MID$(A$(PE),3,1)
3290 LET X=ASC(X$)-64
3300 LET Y=ASC(Y$)-64
3310 CLS#2
3320 IF LEFT$(A$(PE),1)="D" THEN GOTO 3480
3330 FOR L=X TO X+LEN(AN$(PE))-1
3340 FOR H=1 TO 8
3350 IF C$(L,Y-H)="*" THEN GOTO 3380
3360 IF C$(L,Y-H)="0" THEN GOTO 3430
3370 NEXT H
3380 FOR HH=1 TO 8
3390 IF C$(L,Y+HH)="*" THEN GOTO 3440
3400 IF C$(L,Y+HH)="0" THEN GOTO 3430
3410 NEXT HH
3420 GOTO 3440
3430 LET C$(L,Y)="0"
3440 IF H=1 AND HH=1 THEN LET C$(L,Y)="0"
3450 NEXT L
3460 GOSUB 2060
3470 GOTO 1080
3480 FOR L=Y TO Y+LEN(AN$(PE))-1
3490 FOR H=1 TO 8
3500 IF C$(X-H,L)="*" THEN GOTO 3530
3510 IF C$(X-H,L)="0" THEN GOTO 3580
3520 NEXT H
3530 FOR HH=1 TO 8
```

```
3540 IF C$(X+HH,L)="*" THEN GOTO 3590
3550 IF C$(X+HH,L)="0" THEN GOTO 3580
3560 NEXT HH
3570 GOTO 3590
3580 LET C$(X,L)="0"
3590 IF H=1 AND HH=1 THEN LET C$(X,L)="0"
3600 NEXT L
3610 GOSUB 2060
3620 GOTO 1080
3630 REM**TIMER**
3640 LET SEC=SEC-1:IF SEC=-1 THEN LET SEC=59:LET M
I=MI-1
3650 PEN#3,1:LOCATE#3,3,3:PRINT#3," ";USING"##";MI
;:PRINT#3,":";:PRINT#3,USING "##";SEC
3660 IF SEC<10 THEN LOCATE#3,7,3:PRINT#3,"0"
3670 IF MI=0 AND SEC=0 THEN LET D=REMAIN(1):LET GO
VER=1
3680 RETURN
3690 CLS#2:LOCATE#2,7,2:PRINT#2,"OUT OF TIME"
3700 FOR I=1 TO 13
3710 FOR J=1 TO 13
3720 LET C$(I,J)=COPY$(I,J)
3730 NEXT J
3740 NEXT I
3750 GOSUB 2060
3760 CLS#2
3770 PEN#2,3:LOCATE#2,6,2:PRINT#2,"PLAY AGAIN ?"
3780 LET Z$=INKEY$:IF Z$="" THEN GOTO 3780
3790 IF Z$="Y" THEN GOTO 170
3800 IF Z$="N" THEN PAPER 2:PEN 1:LOCATE 1,24:END
3810 GOTO 3780
3820 REM **START**
3830 BORDER 1:INK 0,21:INK 1,0:INK 2,21
3840 PAPER 0
3850 CLS
3860 LOCATE 15,2:PRINT"WELCOME  TO"
3870 LOCATE 12,4:INK 3,6:PEN 3:PRINT"CROSSWORD PUZ
ZLER"
3880 PEN 1
3890 LOCATE 1,8
3900 PRINT" THE OBJECT OF THIS GAME IS TO COMPLETE
 A CROSSWORD THAT WILL BE DISPLAYED ON   THE SCREE
N."
3910 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."
3920 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 '-'.
3930 LOCATE 8,22:PRINT"ENTER DATABASE CASSETTE ":R
ETURN
```

6. Wordsearch

Many newsagents stock puzzle magazines, and of the problems in such publications the favourite of many is the wordsearch. Here we present a computer version of this puzzle which, in conjunction with EDBASE, will provide you with an infinite number of wordsearches to instruct you in the correct spelling of many words.

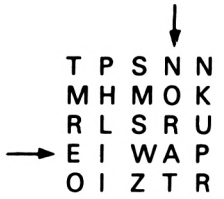
Instructions

The user is required to locate eighteen words hidden in a 16 x 16 matrix of letters. To increase the difficulty of this task, the words can appear backwards or diagonally as well as in their normal orientation. To place the user under some pressure there is a timer shown on the screen and all the words must be located before it reaches zero.

When the program is executed the screen will show the matrix of characters with each position being occupied by an asterisk. Next a message will appear requesting you to place the database cassette into the Datacorder so that the words may be transferred into the computer's memory. The length of time this takes depends on the size of the database.

When the internal buffering is complete the message 'DESIGNING PUZZLE' is shown while the computer selects eighteen words at random and distributes them about the matrix. After these have been chosen, random letters are placed in all the other locations and the board is displayed. The words you are to find are then shown in blue on the right-hand side of the screen and the timer displayed above them is started.

If you think you have correctly located a word in the matrix then you must tell the computer the starting point and the finishing point. This is achieved by positioning a pair of arrows to indicate the row and column in which the key letters occur. For example, to specify the start of the word ASH in the following, the arrows must be positioned as shown:



To indicate the end of the word they should be repositioned to point to the letter H. The horizontal arrow is blue, the vertical arrow red, and the message prompting you to position them uses the same colour code. To move the arrows the cursor keys are used and to make the selection the COPY key should be pressed.

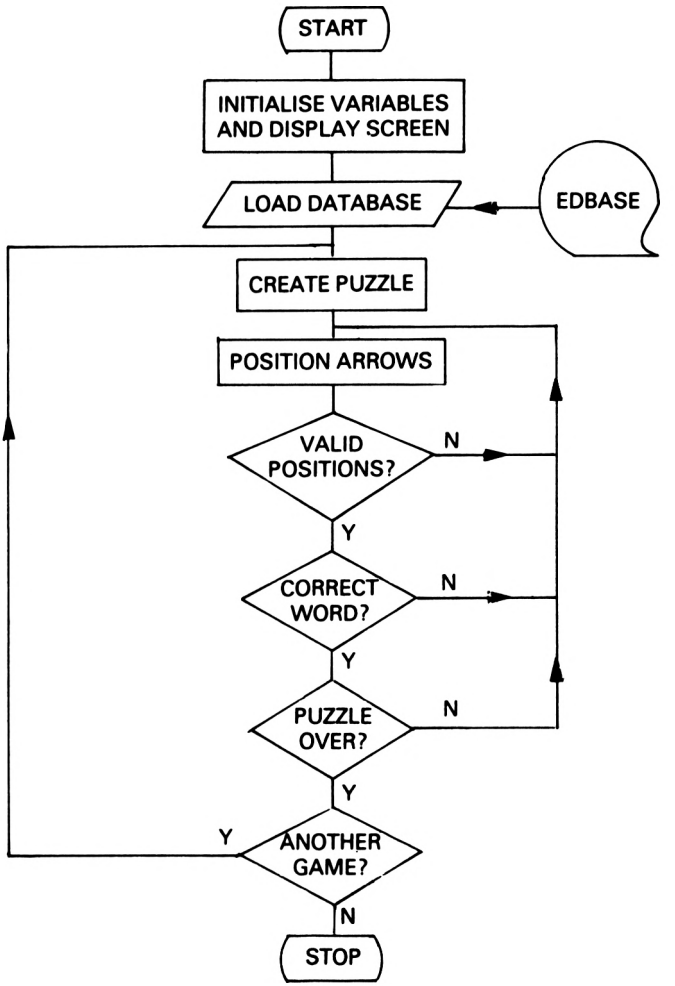
If your choice of starting point and finishing point are legal then the characters in between are highlighted in blue and the computer checks to see if the word is one of those which you are seeking. If it is, then the word in the list is converted to red and if not a message of condolence is shown before the letters in the matrix return to black.

The game ends when you locate all the words or the timer reaches zero, when you are given the choice of having another game or finishing the program.

The program

Flowchart

The flowchart opposite represents the general operation of the program.


START
INITIALISE VARIABLES AND DISPLAY SCREEN
LOAD DATABASE
EDBASE
CREATE PUZZLE
POSITION ARROWS
VALID POSITIONS?
N
Y
CORRECT WORD?
N
Y
PUZZLE OVER?
N
Y
ANOTHER GAME?
Y
N
STOP

Variables

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

TI	Timer value
A$()	Words from database
WORD$()	Words selected for the puzzle
WS()	Used to indicate if the word has been found
FT	Indicates the first time the game is played
B$()	The letters in the matrix
GO	Indicates if the game is over
X,Y	Position of arrows
XS,YS	Starting position of guess
XF,YF	Finishing position of guess
LEGAL	Indicates if these positions are possible
P	Pen colour
AG$	Variable to allow another go
NW	The number of words
D	Direction of word (1-8)
Z$	The characters selected by the user from the matrix

Comments on the program

It is necessary for the program to be highly structured so that several of the control routines can be used many times. A description of these routines is given below.

Lines 10-120
This is the initial routine to dimension the arrays and initialise the variables, displaying the screen with the matrix full of asterisks.

Line 130
If the variable FT is 1 then it is the first time the program has been used and the computer needs to read in the database. Since this only needs to be done once, FT is then given the value 0.

Lines 140-190
The conclusion of the starting process in which the routines to select the words and display them on the screen are used. In line 180, the EVERY command is employed to create a timer giving the game an extra dimension of excitement.

Lines 200-410
These are the lines which display the prompts for the movement of the arrows and call the routines which perform these tasks.

Lines 420-430
If the starting and finishing points are impossible then an error message is printed and you must try again.

Lines 440-500
The letters selected are highlighted and the computer checks to see if they form one of the required words. If they do then the word is changed to a different colour. Also in these lines the computer checks to see if all of the words have been found and if they have not it returns control back to line 200.

Lines 510-620
The puzzle has been correctly solved and a message to this effect is displayed before you are offered another go. The REMAIN(2) command is used in line 510 to stop the timer which is controlled by an EVERY command so that the time remaining no longer decreases.

Lines 630-820
The first subroutine used to create the screen display.

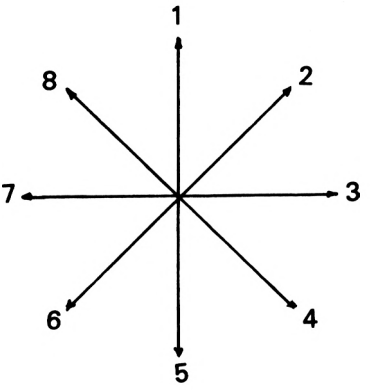
Lines 830-880
This is the routine to print the letters in the matrix on the screen.

Lines 890-1030
This section of code controls the movement of the blue, horizontal arrow. It is in this and the following routines that the test is made to decide if the timer is at zero. This cannot be done in the timer subroutine as control must return from the subroutine called by the EVERY command if it is to be used again.

Lines 1040-1190
A similar routine to the above which controls the movement of the red, vertical arrow.

Lines 1200-1330
This section of program is used to read in the words from the database.

Lines 1340-1690
This is the complex routine which selects a random word and decides if it can be positioned in the matrix. The word can be written in one of eight different directions, determined by the value of D as shown below:



Lines 1700-1760
These lines fill the remaining spaces in the matrix with random letters.

Lines 1770-1820
A short routine to display on the screen the words which are hidden in the matrix.

Lines 1830-2020
This routine identifies the group of characters you have selected from the matrix and forms them into the word Z$. It also contains the code to recolour the chosen characters.

Lines 2030-2110
Here the words in the array WORD$() are examined to see if any of them coincide with the word in Z$.

Lines 2120-2180
A short routine to assess whether the starting and finishing co-ordinates are plausible.

Lines 2190-2230
The routine to control the timer and display an updated value on the screen.

Lines 2240-2290
The routine which is used when the puzzle is not solved in the given amount of time.

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 LET TI=60
20 DIM A$(1001)
30 DIM WORD$(18)
40 DIM WS(18)
50 FT=1
60 DIM B$(16,16)
70 GOSUB 630
80 FOR I=1 TO 16
90 FOR J=1 TO 16
100 LET B$(I,J)="*"
110 NEXT J,I
120 GOSUB 760
130 IF FT=1 THEN LET FT=0:GOSUB 1200
140 FOR I=1 TO 18:LET WS(I)=0:NEXT I
150 GOSUB 1340
160 GOSUB 830
170 GOSUB 1770
180 EVERY 1500,2 GOSUB 2190
190 LET GO=0
200 PEN #4,2:PRINT #4," POSITION BLUE  ARROW  AT S
TART     OF WORD"
210 PEN #4,1:PRINT #4," COPY TO FINISH";
220 GOSUB 890
230 CLS #4
240 PEN #4,3:PRINT #4," POSITION  RED  ARROW  AT S
TART     OF WORD"
```

```
250 PEN #4,1:PRINT #4," COPY TO FINISH";
260 GOSUB 1040
270 LET XS=X:LET YS=Y
280 FOR I=1 TO 2000:NEXT I
290 CLS #5:CLS #6
300 CLS #4
310 PEN #4,2:PRINT #4," POSITION BLUE  ARROW AT FI
NISH     OF WORD"
320 PEN #4,1:PRINT #4," COPY TO FINISH";
330 GOSUB 890
340 CLS #4
350 PEN #4,3:PRINT #4," POSITION  RED  ARROW AT FI
NISH     OF WORD"
360 PEN #4,1:PRINT #4," COPY TO FINISH";
370 GOSUB 1040
380 CLS #4
390 LET XF=X:LET YF=Y
400 FOR I=1 TO 2000:NEXT I
410 CLS #5:CLS #6
420 GOSUB 2120
430 IF LEGAL=0 THEN PEN #4,1:CLS #4:PRINT #4:PRINT
 #4," ILLEGAL  ENTRY":FOR I=1 TO 2000:NEXT I:CLS #
4:GOTO 200
440 LET P=2:GOSUB 1830
450 GOSUB 2030
460 FOR JJ=1 TO 2000:NEXT JJ
470 LET P=1:GOSUB 1830
480 FOR I=1 TO 18
490 IF WS(I)=0 THEN GOTO 200
500 NEXT I
510 LET Z=REMAIN(2)
520 CLS #4
530 PEN #4,2:PRINT #4," PUZZLE  SOLVED":PRINT#4:PE
N #4,3:PRINT #4,"   WELL DONE":PEN #4,1
540 PRINT #4,"  ANOTHER GO?";
550 LET AG$=INKEY$:IF AG$="" THEN GOTO 550
560 IF AG$="N" THEN CLS:END
570 IF AG$<>"Y" THEN GOTO 550
580 LET TI=1
590 CLS #3:CLS #4
600 GOSUB 800
610 LET TI=60
620 GOTO 80
630 REM ** DESIGN BOARD **
640 INK 0,18:INK 1,0:INK 2,2:INK 3,6
650 BORDER 18:CLS
660 WINDOW #1,3,18,3,18:PAPER #1,0:PEN #1,1:CLS #1
670 WINDOW #2,26,35,3,3:PAPER #2,0:PEN #2,3:CLS #2
680 WINDOW #3,22,38,6,24:PAPER #3,0:PEN #3,1:CLS #
3
```

```
690 WINDOW #4,3,18,21,24:PAPER #4,0:PEN #4,1:CLS #
4
700 WINDOW #5,3,18,1,1:PAPER #5,0:PEN #5,2:CLS #5
710 WINDOW #6,1,1,3,18:PAPER #6,0:PEN #6,3:CLS #6
720 PLOT 376,372,1:DRAW 376,348:DRAW 568,348:DRAW
568,372:DRAW 376,372
730 PLOT 326,8,1:DRAW 326,326:DRAW 616,326:DRAW 61
6,8:DRAW 326,8
740 PLOT 24,372,1:DRAW 294,372:DRAW 294,104:DRAW 2
4,104:DRAW 24,372
750 PLOT 24,84:DRAW 294,84:DRAW 294,8:DRAW 24,8:DR
AW 24,84
760 FOR I=1 TO 16
770 FOR J=1 TO 16
780 LOCATE #1,I,J:PRINT #1,B$(I,J);
790 NEXT J,I
800 PRINT #2,"TIMER  60";
810 PEN #3,1:LOCATE #3,7,1:PRINT #3,"WORDS"
820 RETURN
830 REM ** DISPLAY BOARD **
840 FOR I=1 TO 16
850 FOR J=1 TO 16
860 LOCATE #1,I,J:PRINT #1,B$(I,J);
870 NEXT J,I
880 RETURN
890 REM ** TOP ARROW **
900 LET X=1
910 PRINT #5,CHR$(241);
920 LET M$=INKEY$
930 IF M$="" THEN GOTO 920
940 IF TI=0 THEN GOTO 2240
950 IF ASC(M$)=224 THEN RETURN
960 IF ASC(M$)=243 THEN X=X+1:GOTO 990
970 IF ASC(M$)=242 THEN X=X-1:GOTO 990
980 GOTO 920
990 IF X<1 THEN LET X=1
1000 IF X>16 THEN LET X=16
1010 CLS #5
1020 LOCATE #5,X,1:PRINT #5,CHR$(241);
1030 GOTO 920
1040 REM ** SIDE ARROW **
1050 LET Y=1
1060 PRINT #6,CHR$(243);
1070 LET M$=INKEY$
1080 IF M$="" THEN GOTO 1070
1090 IF TI=0 THEN GOTO 2240
1100 IF ASC(M$)=224 THEN RETURN
1110 IF ASC(M$)=241 THEN Y=Y+1:GOTO 1150
1120 IF M$="" THEN GOTO 1110
1130 IF ASC(M$)=240 THEN Y=Y-1:GOTO 1150
```

```
1140 GOTO 1070
1150 IF Y<1 THEN Y=1
1160 IF Y>16 THEN Y=16
1170 CLS #6
1180 LOCATE #6,1,Y:PRINT #6,CHR$(243);
1190 GOTO 1070
1200 REM ** INPUT DATABASE **
1210 PEN #4,3:PRINT #4,"LOADING DATABASE":PEN #4,2
:PRINT #4," INPUT CASSETTE  AND PRESS PLAY";:PEN #
4,1
1220 LET I=1
1230 OPENIN "!EDBASE"
1240 INPUT#9,A$(I)
1250 IF LEN(A$(I))=1 THEN GOTO 1240
1260 IF A$(I)="XXX" THEN GOTO 1300
1270 INPUT#9,A$
1280 LET I=I+1
1290 GOTO 1240
1300 LET NW=I-1
1310 CLOSEIN
1320 CLS #4
1330 RETURN
1340 REM ** CHOOSE WORDS **
1350 CLS#4:PEN #4,2:PRINT #4,"DESIGNING PUZZLE";
1360 PRINT #4:PRINT #4,"  PLEASE WAIT";
1370 FOR II=1 TO 18
1380 LET N=1+INT(RND(1)*NW)
1390 FOR J=1 TO II
1400 IF WORD$(J)=A$(N) THEN GOTO 1380
1410 NEXT J
1420 LET X=1+INT(RND(1)*16):LET Y=1+INT(RND(1)*16)
:LET D=1+INT(RND(1)*8)
1430 IF D=1 AND Y-LEN(A$(N))<1 THEN GOTO 1420
1440 IF D=2 AND (Y-LEN(A$(N))<1 OR X+LEN(A$(N))>16
) THEN GOTO 1420
1450 IF D=3 AND X+LEN(A$(N))>16 THEN GOTO 1420
1460 IF D=4 AND (Y+LEN(A$(N))>16 OR X+LEN(A$(N))>1
6) THEN GOTO 1420
1470 IF D=5 AND Y+LEN(A$(N))>16 THEN GOTO 1420
1480 IF D=6 AND (Y+LEN(A$(N))>16 OR X-LEN(A$(N))<1
) THEN GOTO 1420
1490 IF D=7 AND X-LEN(A$(N))<1 THEN GOTO 1420
1500 IF D=8 AND (Y-LEN(A$(N))<1 OR X-LEN(A$(N))<1)
 THEN GOTO 1420
1510 LET XX=X:LET YY=Y
1520 FOR I=1 TO LEN(A$(N))
1530 IF B$(XX,YY)="*" THEN GOTO 1550
1540 IF B$(XX,YY)<>MID$(A$(N),I,1) THEN GOTO 1380
1550 IF D=1 OR D=2 OR D=8 THEN YY=YY-1
1560 IF D=4 OR D=5 OR D=6 THEN YY=YY+1
```

```
1570 IF D=2 OR D=3 OR D=4 THEN XX=XX+1
1580 IF D=6 OR D=7 OR D=8 THEN XX=XX-1
1590 NEXT I
1600 LET XX=X:LET YY=Y
1610 FOR I=1 TO LEN(A$(N))
1620 LET B$(XX,YY)=MID$(A$(N),I,1)
1630 IF D=1 OR D=2 OR D=8 THEN YY=YY-1
1640 IF D=4 OR D=5 OR D=6 THEN YY=YY+1
1650 IF D=2 OR D=3 OR D=4 THEN XX=XX+1
1660 IF D=6 OR D=7 OR D=8 THEN XX=XX-1
1670 NEXT I
1680 LET WORD$(II)=A$(N)
1690 NEXT II
1700 REM ** COMPLETE BOARD **
1710 FOR I=1 TO 16
1720 FOR J=1 TO 16
1730 IF B$(I,J)="*" THEN LET B$(I,J)=CHR$(65+INT(R
ND(1)*26))
1740 NEXT J,I
1750 CLS #4
1760 RETURN
1770 REM ** PRINT WORDS **
1780 PEN#3,2
1790 FOR I=1 TO 18
1800 LOCATE#3,1,I+1:PRINT#3, WORD$(I)
1810 NEXT I
1820 RETURN
1830 REM ** FIND POSITION OF WORD **
1840 LET Z$=""
1850 IF XS=XF AND YF>YS THEN LET D=5
1860 IF XS=XF AND YF<YS THEN LET D=1
1870 IF YS=YF AND XF>XS THEN LET D=3
1880 IF YS=YF AND XF<XS THEN LET D=7
1890 IF XF-XS=YS-YF AND XF>XS THEN LET D=2
1900 IF XF-XS=YF-YS AND XF>XS THEN LET D=4
1910 IF XF-XS=YS-YF AND XF<XS THEN LET D=6
1920 IF XF-XS=YF-YS AND XF<XS THEN LET D=8
1930 LET XX=XS:LET YY=YS
1940 PEN#1,P
1950 LOCATE#1,XX,YY:PRINT#1,B$(XX,YY);
1960 LET Z$=Z$+B$(XX,YY)
1970 IF XX=XF AND YY=YF THEN RETURN
1980 IF D=8 OR D=1 OR D=2 THEN LET YY=YY-1
1990 IF D=4 OR D=5 OR D=6 THEN LET YY=YY+1
2000 IF D=2 OR D=3 OR D=4 THEN LET XX=XX+1
2010 IF D=6 OR D=7 OR D=8 THEN LET XX=XX-1
2020 GOTO 1950
2030 REM ** CHECK WORD **
2040 FOR I=1 TO 18
2050 IF WORD$(I)=Z$ THEN GOTO 2090
```

```
2060 NEXT I
2070 CLS #4:PEN #4,1:LOCATE #4,5,2:PRINT #4,"BAD L
UCK":FOR I=1 TO 2000:NEXT I:CLS #4:RETURN
2080 FOR J=1 TO 2000:NEXT J:GOTO 2110
2090 LOCATE#3,1,I+1:PEN#3,3:PRINT#3,Z$
2100 LET WS(I)=1
2110 RETURN
2120 REM ** LEGALITY CHECK **
2130 LET LEGAL=0
2140 IF XS=XF AND YS=YF THEN LEGAL=0:RETURN
2150 IF XS=XF THEN LEGAL=1
2160 IF YS=YF THEN LEGAL=1
2170 IF ABS(XF-XS)=ABS(YF-YS) THEN LEGAL=1
2180 RETURN
2190 REM ** TIMER ROUTINE **
2200 IF TI=0 THEN RETURN
2210 LET TI=TI-1
2220 LOCATE#2,8,1:PRINT#2,USING "##";TI;
2230 RETURN
2240 REM ** OUT OF TIME **
2250 CLS #5:CLS #6
2260 LET Z=REMAIN(2)
2270 CLS #4:PEN #4,1:PRINT #4,"  OUT OF TIME"
2280 PRINT #4:PRINT #4,"  ANOTHER GO?"
2290 GOTO 550
```

7. Horseracing

This is one of the few programs in this book which does not employ the database, but draws its educational value from the user learning to estimate the sum of a group of numbers. There are five levels of play with level 1 being the easiest, and up to five people can play.

As the game progresses, your horse will move by a distance which depends on the accuracy of your estimate. The maximum you can move is 100 places and the minimum is 0.

Instructions

When executed the screen will show the track, the five horses and the crowd which has assembled to enjoy an afternoon's sport. The computer will ask you what level you require and the response should be a number in the range 1 to 5 (1 is the easiest and 5 the most difficult).

Next the computer will enquire about the number of players and again you will have the choice between 1 and 5. After this you are asked to input their names, and then a brief résumé of the instructions is displayed.

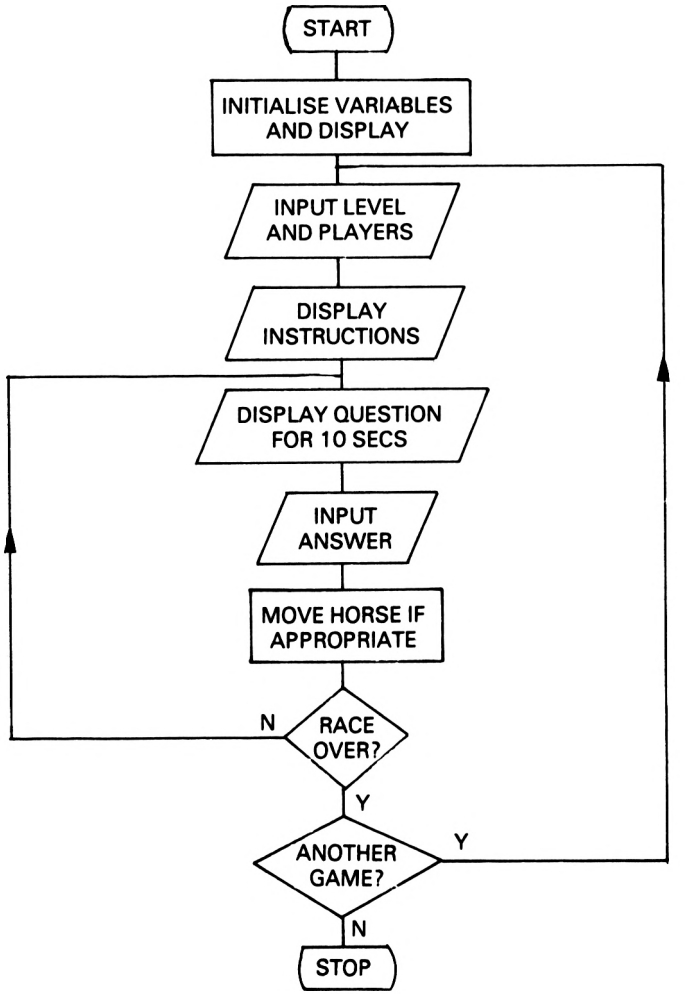
The game now begins with each player in turn being shown a mathematical problem for which they must estimate the answer. After ten seconds the problem disappears. Although there is now unlimited time in which to decide on an answer, it is very difficult to retain enough information in your memory to use this time effectively.

The race is won by the first person to reach the finishing post and then the computer offers you the choice of having another game or putting an end to the day's racing.

The program

Flowchart

The flowchart 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.

PS(n)	Position of horse n
MO	Current player's move
PL	Total number of players
N$()	Players' names
LEV$	The character representing the level
LEV	The numerical equivalent of LEV$
PL$	The character representing the number of players
AG$	Another game?
INS$	Contains a line of instructions
A()	Contains the seven numbers to be summed
ANS$	The answer
ANS	The numerical equivalent of ANS$
AM	Amount to move by
WT()	Array holding the music

Comments on the program

The program makes full use of subroutines and these are clearly marked throughout the listing.

Lines 10-140
The main program, which calls the necessary subroutines during the game.

Lines 150-470
The initial routine, which defines the colours and the windows and draws the horses and people in the grandstand.

Lines 480-900
This is where the level of play is chosen and the number of players and their names are input. If it is the first time that the race is to be held then the instructions are displayed.

Lines 910-1120
This routine constructs the arithmetic problem by producing seven random integers which must be added. The numbers which the computer chooses depend on the level which has already been selected in another subroutine.

The program allows you to input your answer after displaying the sum for ten seconds and then calculates the amount by which your horse should move.

Lines 1130-1240
This section of program moves the appropriate horse by the amount specified in the previous routine.

Lines 1250-1330
A section of code which is used when the game has finished to display the winner and offer another race.

Lines 1340-1500
This is the final routine which is used for the graphics at the end of the day's racing.

Lines 1510-1560
A short routine to create the user-defined graphics used for the railings, winning post and horses.

Lines 1570-1650
This section holds the data for the notes in the jingle which is played at the start of a player's move.

Lines 1660-1720
The section of program which plays the music read in the above routine.

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 GOSUB 1510
20 GOSUB 150
30 GOSUB 480
40 GOSUB 1570
50 FOR MO=1 TO PL
60 GOSUB 1660
70 WHILE SQ(1)>127
80 WEND
90 GOSUB 910
100 NEXT MO
110 GOTO 50
120 FOR Z=1 TO 5:LET PS(Z)=40:NEXT Z
130 IF INKEY$="" THEN GOTO 130:MODE 1
140 END
150 REM ** INITIAL ROUTINE **
160 MODE 0
170 INK 0,18:INK 1,0:INK 2,26:INK 3,2:INK 4,6:INK
5,20:INK 6,0:INK 7,26:INK 8,6:INK 9,24:INK 10,2:IN
K 13,24:INK 15,9:INK 14,10
180 PAPER 2:CLS
190 WINDOW #1,1,20,9,18:PAPER #1,0:CLS #1
200 WINDOW #7,1,20,10,16:PAPER #1,0:CLS #1
210 WINDOW #2,1,20,4,8:PAPER #2,3:PEN #2,1:CLS #2
220 WINDOW #3,1,20,9,9:PAPER #3,0:CLS #3
230 WINDOW #4,1,20,17,17:PAPER #4,0:CLS #4
240 WINDOW #5,1,20,1,3:PAPER #5,5:PEN #5,4:CLS #5:
PRINT #5:PRINT #5,"    HORSE RACING"
250 PLOT 0,399,1:DRAWR 638,0:DRAWR 0,-46:DRAWR -64
0,0:DRAWR 0,46
260 BORDER 2
270 WINDOW #6,1,20,19,25:PAPER #6,13:CLS #6:PEN #6
,8
280 PEN #2,13:LOCATE #2,5,2:PRINT #2,CHR$(231):PEN
 #2,1
290 FOR J=1 TO 20
300 FOR I=3 TO 4
310 LOCATE #I,J,1:PRINT #I,CHR$(158);
320 NEXT I,J
330 LOCATE #2,20,5:PRINT #2,CHR$(250);:LOCATE #3,2
0,1:PRINT #3,CHR$(251);
340 LET K=48
350 FOR J=272 TO 320 STEP 2
360 FOR I= 450-K TO 550+K STEP 2
370 PLOT I,J,INT(16*RND(1))
380 NEXT I
390 LET K=K-2
400 NEXT J
410 FOR Z=10 TO 6 STEP -1
420 PLOT 640,400,Z
430 MOVE -20,20*Z+44
```

```
440 TAG:PRINT 11-Z;CHR$(252);
450 TAGOFF
460 NEXT Z
470 RETURN
480 REM ** INPUT PLAYERS **
490 DIM N$(5),A(7),PS(5)
500 FOR Z=1 TO 5:LET PS(Z)=44:NEXT Z
510 PRINT #6:PRINT #6," ENTER LEVEL(1-5) ";
520 LET LEV$=INKEY$:IF LEV$<"1" OR LEV$>"5" THEN G
OTO 520
530 PRINT #6,LEV$:LET LEV=VAL(LEV$)
540 PRINT #6:PRINT #6," HOW MANY PLAYERS ";
550 LET PL$=INKEY$:IF PL$<"1" OR PL$>"5" THEN GOTO
 550
560 PRINT #6,PL$
570 LET PL=ASC(PL$)-48
580 FOR Z=1 TO 1000:NEXT Z
590 CLS #6:PRINT #6
600 PRINT #6,"INPUT NAMES: "
610 FOR I=1 TO PL
620 LOCATE #6,13,I+1:PRINT #6,USING"#";I:LOCATE #6
,15,I+1:LINE INPUT #6,N$(I)
630 IF LEN(N$(I))>5 THEN LET N$(I)=LEFT$(N$(I),5)
640 NEXT I
650 FOR Z=1 TO 1000:NEXT Z
660 CLS #6
670 IF AG$="Y" THEN CLS#7:GOTO 410
680 LOCATE #6,1,6
690 FOR I=1 TO 11
700 READ INS$
710 FOR Z=1 TO 1000:NEXT Z
720 PRINT #6,INS$
730 NEXT I
740 FOR I=1 TO 3
750 PRINT #6:PRINT #6
760 FOR Z=1 TO 1000:NEXT Z
770 NEXT I
780 CLS #6
790 DATA EACH  PLAYER  HAS  A
800 DATA TURN  AT  ESTIMATING
810 DATA THE  ANSWER  TO  A
820 DATA MATHEMATICAL PROBLEM
830 DATA IN TEN SECONDS.
840 DATA
850 DATA
860 DATA THE MOVEMENT  OF HIS
870 DATA HORSE  IS  DEPENDENT
880 DATA ON THE  ACCURACY  OF
890 DATA THE ESTIMATE.
900 RETURN
```

```
910 REM ** PROBLEM **
920 CLS #6
930 FOR I=1 TO 7
940 LET A(I)=1+INT(RND(1)*(20*LEV-1)):IF A(I)<10 T
HEN GOTO 940
950 NEXT I
960 PRINT #6:PRINT #6,N$(MO):PRINT #6
970 FOR I=1 TO 7
980 PRINT #6,USING"##";A(I);
990 IF I=7 THEN GOTO 1010
1000 IF I<>7 THEN PRINT #6,"+";
1010 NEXT I
1020 LET TT=TIME
1030 IF TIME-TT<3000 THEN GOTO 1030
1040 LOCATE #6,1,4:PRINT #6,"   ANSWER
  ":LOCATE #6,12,4
1050 LINE INPUT #6,ANS$
1060 LET ANS=VAL(ANS$)
1070 LOCATE #6,1,6
1080 LET DIFF=INT(A(1)+A(2)+A(3)+A(4)+A(5)+A(6)+A(
7)-ANS):IF DIFF<0 THEN LET DIFF=DIFF*(-1)
1090 LET AM=2*(50-DIFF):IF AM<0 OR AM>100 THEN AM=
0
1100 IF AM=0 THEN PRINT #6,"BAD LUCK. NO MOVE":FOR
 Z=1 TO 1000:NEXT Z
1110 IF AM<>0 THEN PRINT #6,"MOVE";AM;"PLACES":GOS
UB 1130
1120 RETURN
1130 REM ** MOVE HORSE **
1140 FOR Q=PS(MO) TO PS(MO)+AM
1150 PLOT 640,400,11-MO
1160 MOVE Q,20*(11-MO)+44
1170 TAG
1180 PRINT " ";CHR$(252);
1190 IF Q=360 THEN INK 14,0,26:INK 15,2,6
1200 IF Q=560 THEN GOTO 1250
1210 NEXT Q
1220 TAGOFF
1230 LET PS(MO)=PS(MO)+AM
1240 RETURN
1250 REM ** GAME OVER **
1260 CLS #6
1270 PRINT #6:PRINT #6,"   ";N$(MO);" HAS WON !"
1280 FOR Z=1 TO 200
1290 INK 11-MO,1+INT(RND(1)*16)
1300 NEXT Z
1310 INK 4,6:INK 5,20:INK 6,0:INK 7,26:INK 8,6:INK
 9,24:INK 10,2:INK 15,9:INK 14,10
1320 PRINT #6:PRINT #6:INPUT #6," ANOTHER GO(Y/N)"
;AG$
```

```
1330 IF AG$="Y" THEN CLS #6:GOSUB 500:GOTO 50
1340 REM ** END OF DAY **
1350 CLS #7
1360 PAPER #6,3:CLS #6
1370 LET K=0
1380 FOR J=320 TO 272 STEP -2
1390 FOR I= 450-K TO 550+K STEP 2
1400 IF J=320 AND I=450 THEN FOR Z=600 TO 614 STEP
 2:PLOT Z,272,1+INT(RND(1)*15):NEXT Z
1410 IF J=320 AND I=450 THEN FOR Z=630 TO 638 STEP
 2:PLOT Z,272,1+INT(RND(1)*15):NEXT Z
1420 PLOT I,J,1
1430 NEXT I
1440 LET K=K+2
1450 NEXT J
1460 FOR Z=600 TO 614 STEP 2:PLOT Z,272,3:NEXT Z
1470 FOR Z=630 TO 638 STEP 2:PLOT Z,272,3:NEXT Z
1480 PEN #2,2:LOCATE #2,5,2:PRINT #2,CHR$(41):PEN
#2,1
1490 IF INKEY$="" THEN GOTO 1490
1500 END
1510 REM ** USER DEFINED GRAPHICS **
1520 SYMBOL AFTER 250
1530 SYMBOL 250,0,&3C,&66,&C3,&C3,&66,&3C,&18
1540 SYMBOL 251,&18,&18,&18,&FF,&FF,&18,&18,&18
1550 SYMBOL 252,&4,&6,&7F,&FD,&FC,&DE,&C5,&A5
1560 RETURN
1570 REM ** MUSIC **
1580 RESTORE 1630
1590 DIM WT(100)
1600 FOR I=1 TO 45
1610 READ WT(I)
1620 NEXT I
1630 DATA 159,478,190,159,506,190,190,568,239,159,
638,190
1640 DATA 142,716,213,159,638,190,190,568,239,190,
638,239,213,716,284,179,851,284,190,638,319,213,63
8,358,239,956,379,239,956,379,239,956,379
1650 RETURN
1660 REM ** PLAY TUNE **
1670 FOR I=1 TO 43 STEP 3
1680 SOUND 1,WT(I),20
1690 SOUND 2,WT(I+1),20
1700 SOUND 4,WT(I+2),20
1710 NEXT I
1720 RETURN
```

8. Nightmare Park

Nightmare Park must be the only adventure game to be unsolvable yet have a cult following amongst ardent adventurers. The object of the game is to get from one end of a maze to the other without falling foul of the many hazards which are likely to be encountered. This educational version of the game incorporates questions on numerous subjects and you will need some luck as well as a great deal of skill if you are to survive and escape from the maze.

Instructions

When the program is executed the database will be loaded (with the usual delay) and the screen will be divided into two sections with the top window containing a map of the maze and the bottom window containing reports and the questions.

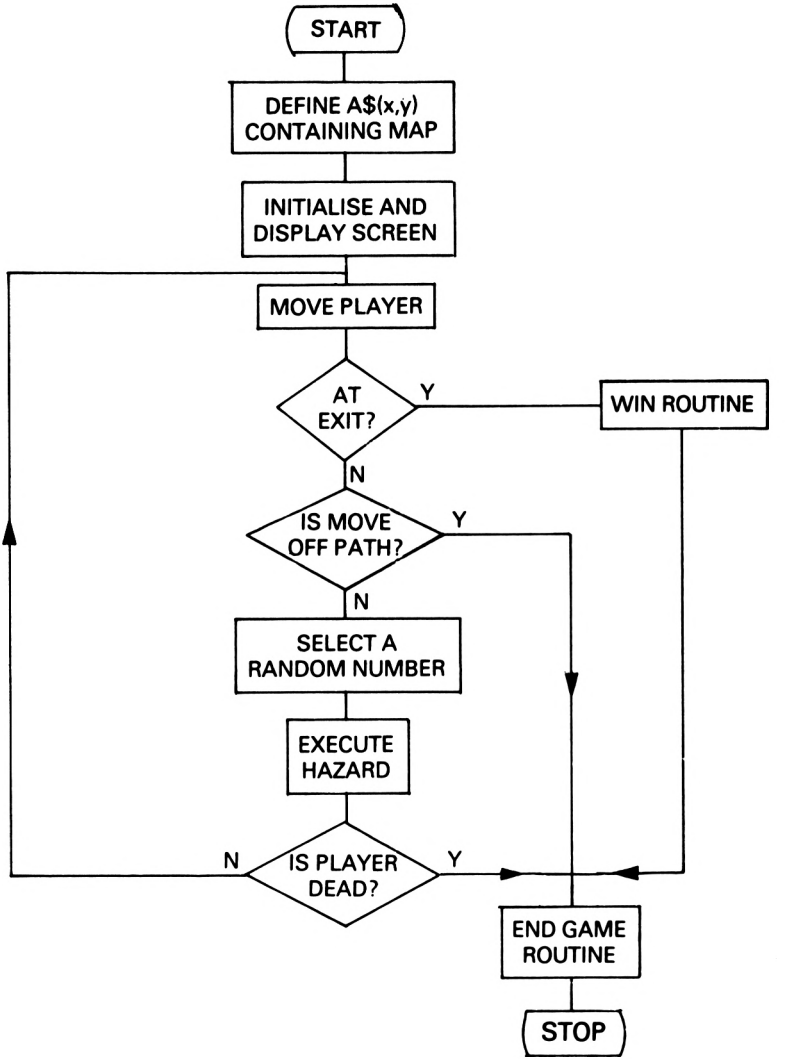
Movement around the maze is achieved by using the cursor keys. After each move a hazard is likely to be encountered and this will be displayed in the bottom window.

As this is an adventure program no more instructions are given and it is up to you to learn as you play the game.

The program

Flowchart

The flowchart on p.86 represents the general operation of the program.


START
DEFINE A$(x,y) CONTAINING MAP
INITIALISE AND DISPLAY SCREEN
MOVE PLAYER
AT EXIT?
Y
WIN ROUTINE
N
IS MOVE OFF PATH?
Y
N
SELECT A RANDOM NUMBER
EXECUTE HAZARD
N
IS PLAYER DEAD?
Y
END GAME ROUTINE
STOP

Variables

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

WORD$()	Array used to store individual letters of a word
A$(X,Y)	The map
X,Y	Co-ordinates of the player within the maze
TOT	Total used in mathematical question
QU$	Mathematical question
DEAD	A flag used to indicate the end of a game
D$()	Array to hold numerical sequence
G$	Word used in the anagram
F$	The anagram
N1,N2,N3	Number of words used in 'odd-one-out'.
XW,XY	Co-ordinates for word flashed on screen
V	Number of words in database

Comments on the program

The program is structured, and the following notes give details of each of the individual routines.

Lines 10-25
Define the array containing the map.

Lines 30-150
Contain the data for the map.

Lines 160-230
This routine defines the inks and initialises the screen display.

Lines 1000-1020
This is the main routine that calls all the subroutines relating to the various hazards.

Lines 2000-8530
These lines contain the nine routines pertaining to the various hazards and are called from the main routine depending on the result of a random integer in the range 1–9.
These routines are as follows:

2000	The mad mathematician
3000	The memory man
4000	Sequencer
5000	Anagrams
6000	Odd one out
7000	Capital cities
7500	The flasher
8000	The asterisk
8500	No hazard

Lines 9000-9230
This routine controls the movement of the player around the maze.

Lines 9600-9690
A routine used to input the database from cassette into memory.

Lines 9700-9730
A short section of program to clear the lower window after a hazard has been executed.

Lines 9900-9960
The win routine, called in the unlikely event that the player escapes from the maze alive.

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.

```
5 GOSUB 9600
6 DIM WORD$(20)
10 MODE 1
15 IF GO=1 THEN GOTO 60
20 IF GO=0 THEN LET GO=1:DIM A$(13,40)
25 FOR I=1 TO 13:FOR J=1 TO 40:READ A$(I,J):NEXT J
,I
30  DATA *,*,*,*,*,*,*,*,*,*,*,*,*,*,*,*,*,*,*,*,*
,*,*,*,*,*,*,*,*,*,*,*,*,*,*,*,*,*,*,*,*,*,*,*,*
40 DATA *,*,*,*,*,*,*,*,*,*,*,*,*,*,*,*, ,*,*,*,*,*,*,
*,*, ,*,*,*, ,*, , , ,*, , , , , , ,*
50 DATA *,*,*,*,*,*,*,*, , , , , , , , , ,*,*,*,*,*, , ,
```

```
, , , , , , , ,*,*,*,*,*, ,*,*,*,*, ,*
60 DATA *,*,*,*,*,*,*,*,*,*, ,*,*,*, , , , , , , ,*,
*,*,*,*, ,*, ,*,*, ,*,*, ,*,*,*, , ,*
70 DATA *,*,*,*,*,*,*,*,*, , ,*, , , ,*,*,*,*, , ,*,
*,*,*,*, ,*, ,*, , , , , , , , ,*, , ,*
80 DATA *,*,*,*,*, , , , , , , , , , , , , , , , , ,*,
, , , , , ,*, ,*, ,*,*, ,*,*,*,*,*, ,*
90 DATA G, , , , , ,*,*,*,*,*,*,*,*, ,*,*,*,*,*,*,
*,*,*,*,*,*, , , ,*,*, , ,*, , ,*, ,*
100 DATA *,*, ,*,*, ,*,*,*,*,*, , , , ,*,*, , , ,
, , , , , , ,*, , ,*, ,*,*, , , , ,G
110 DATA *,*, ,*,*, , , , , , , , ,*,*, ,*, ,*,*,*,*
,*, ,*,*,*,*, ,*,*,*,*, ,*,*,*,*,*,*,*
120 DATA *, , , , , , ,*,*,*,*,*,*,*,*, ,*, ,*,*,*,*
,*, ,*,*,*,*,*,*, , , , , , ,*, , , , , ,*
130 DATA *,*,*,*,*,*,*,*, , , , , , , , , , , , ,*,*,*,*
,*, ,*,*,*,*,*,*,*,*,*,*, ,*,*,*,*,*, ,*
140 DATA *,*,*,*,*,*,*,*,*,*,*,*,*,*,*, ,*,*,*,*, , ,
, , , , , , , , ,*, , , , , , , , , , , ,*
150 DATA *,*,*,*,*,*,*,*,*,*,*,*,*,*,*,*,*,*,*,*,*,*,*,*
,*,*,*,*,*,*,*,*,*,*,*,*,*,*,*,*,*,*,*
160 BORDER 5
161 INK 1,18:INK 2,9:INK 3,0
162 INK 0,25
163 WINDOW #3,1,40,14,25
165 PAPER#3,0:PEN#3,3:CLS
170 WINDOW #2,1,40,1,13
180 PAPER #2,1:CLS#2:PEN#2,2
185 LOCATE #3,18,4:PRINT#3,"WELCOME":LOCATE #3,13,
6:PRINT#3,"TO NIGHTMARE PARK!"
190 FOR I=1 TO 13:FOR J=1 TO 40:LOCATE#2,J,I:IF A$
(I,J)="G" THEN PRINT#2,CHR$(243); ELSE PRINT#2,A$(
I,J);
195 NEXT J,I
200 Y=7:X=2
205 LOCATE#2,X,Y:PEN#2,3
210 LOCATE#2,X,Y:IF X/2=INT(X/2)THEN PRINT#2,CHR$(
250) ELSE PRINT#2,CHR$(251)
215 CLS#3
220 LOCATE#3,16,4:PRINT#3,"BEWARE!!"
230 LOCATE#3,12,7:PRINT#3,"KEEP TO THE PATH"
1000 GOSUB 9000
1010 LET NUM=INT (RND*10)
1020 ON NUM GOTO 2000,3000,4000,5000,6000,7000,750
0,8000,8500
2000 REM **THE MAD MATHEMATICIAN**
2001 FOR I=1 TO 20:LET Z$=INKEY$:NEXT I
2010 CLS#3
2020 PRINT#3:PRINT#3,"          THE MAD MATHEMATIC
IAN"
```

```
2030 LET TOT=0
2035 LET QU$=""
2040 FOR I=1 TO 5
2050 LET D=INT(RND*20)-10
2060 LET TOT=TOT +D
2070 IF D<0 THEN LET QU$=QU$+STR$(D)
2080 IF D>=0 THEN LET QU$=QU$+"+"+RIGHT$(STR$(D),L
EN(STR$(D))-1)
2090 NEXT I
2095 IF LEFT$(QU$,1)="-"THEN GOTO 2110
2100 LET QU$=RIGHT$(QU$,LEN(QU$)-1)
2110 PRINT#3:PRINT#3,"YOU HAVE TEN SECONDS TO WORK
 OUT:-"
2115 PRINT#3
2120 PRINT#3,QU$
2125 EI
2130 AFTER 500 GOSUB 2500
2135 PRINT#3
2140 INPUT#3,"ANSWER = ";Q$
2150 LET S=REMAIN(0)
2160 LET Q=VAL(Q$)
2170 IF Q<> TOT THEN GOTO 2400
2180 PRINT#3:PRINT#3,"SO YOU THINK YOU´RE SMART EH
?"
2200 GOTO 9700
2400 PRINT#3:PRINT#3,"WRONG YET AGAIN!!!!!"
2410 GOTO 9800
2500 PRINT#3:PRINT#3,"TOO SLOW...BETTER LUCK NEXT
TIME!!":GOTO 9800
2998 STOP
2999 GOTO 1000
3000 REM **THE MEMORY MAN**
3005 FOR I=1 TO 20:LET Z$=INKEY$:NEXT I
3010 CLS#3
3020 PRINT#3:PRINT#3,"          THE MEMORY MAN"
3030 PRINT#3
3040 FOR I=1 TO 6
3050 LET D$(I)=CHR$(INT(RND*26)+65)
3060 NEXT I
3070 PRINT#3,"YOU WILL SEE THE FOLLOWING SEQUENCE
FOR 5 SECONDS..."
3080 PRINT#3
3090 FOR I=1 TO 6
3100 PRINT#3,D$(I);"  ";
3110 NEXT I
3120 EI
3125 LET FLAG=1
3130 AFTER 250 GOSUB 3150

3135 IF DEAD=1 THEN LET DEAD=0:GOTO 9800
```

```
3136 IF DEAD=2 THEN LET DEAD=0:GOTO 9700
3140 GOTO 3135
3150 LOCATE#3,1,6:PRINT#3,"                          "
3151 EI
3160 PRINT#3,"REPEAT THE SEQUENCE..."
3170 FOR I=1 TO 6
3180 LOCATE #3,1,9:PRINT#3,"LETTER.";I;"=";
3190 LINE INPUT#3,T$
3200 IF T$<>D$(I) THEN GOTO 3500
3205 LOCATE#3,12,9:PRINT#3,"       "
3210 NEXT I
3220 PRINT#3:PRINT#3,"YOU HAVE BEATEN THE MEMORY M
AN!!"
3230 LET DEAD=2:RETURN
3500 PRINT#3:PRINT#3,"WHAT AN APPALLING MEMORY!!":
LET DEAD=1:RETURN
3701 FOR I=1 TO 2000:NEXT I
3999 GOTO 1000
4000 REM **THE SEQUENCER**
4001 FOR I=1 TO 20:LET Q$=INKEY$:NEXT I
4005 FOR I=1 TO 20:LET Z$=INKEY$:NEXT I
4010 CLS#3
4020 PRINT#3:PRINT#3,"           THE SEQUENCER"
4030 PRINT#3:PRINT#3,"COMPLETE THE FOLLOWING SEQUE
NCE BY      GIVING THE NEXT TWO NUMBERS..."
4040 LET M=INT(RND*6)+1
4050 LET C=INT(RND*10)+1
4060 PRINT#3
4070 FOR I=1 TO 4
4080 LET U=M*I+C
4090 PRINT#3,U;",";
4095 NEXT I
4100 PRINT#3," - , - "
4110 PRINT#3
4120 PRINT#3,"FIRST NUMBER  = ";:LINE INPUT#3,N1$:
LET N1=VAL(N1$)
4130 IF N1<>M*5+C THEN GOTO 4500
4140 PRINT#3,"SECOND NUMBER = ";:LINE INPUT#3,N2$:
LET N2=VAL(N2$)
4150 IF N2<>N1+M THEN GOTO 4500
4160 PRINT#3:PRINT#3,"WHAT A GENIUS!!":GOTO 9700
4500 PRINT#3:PRINT#3,"NEVER MIND..THERE IS ALWAYS
A NEXT TIME!":GOTO 9800
4999 GOTO 1000
5000 REM **THE ANAGRAM MAN**
5005 FOR I=1 TO 20:LET Z$=INKEY$:NEXT I
5010 CLS#3
5020 PRINT#3:PRINT#3,"              THE ANAGRAM MA
N"
5030 PRINT#3:PRINT#3,"YOU WILL BE GIVEN THIRTY SEC
```

```
ONDS TO     UNRAVEL THE FOLLOWING ANAGRAM."
5040 R=INT(RND*(V-1))+1
5045 F$=""
5050 G$=N$(R)
5060 T=LEN(N$(R))
5070 FOR C=T TO 2 STEP-1
5080 B=INT(RND*C)+1
5090 F$=F$+MID$(G$,B,1)
5100 G$=LEFT$(G$,B-1)+MID$(G$,B+1,LEN(G$)-B)
5110 NEXT C
5120 LET F$=F$+G$
5130 PRINT#3:PRINT#3,F$
5135 AFTER 1500 GOSUB 5200
5140 PRINT#3:INPUT#3,"WORD  ";ANSWER$
5145 IF FLAG=1 THEN GOTO 5210
5150 D=REMAIN(0)
5160 IF ANSWER$<>N$(R) THEN PRINT#3:PRINT#3,"WRONG
, IT WAS ";N$(R);".":GOTO 9800
5170 PRINT#3:PRINT#3,"WOW!! YOU GOT IT RIGHT.":GOT
O 9700
5200 FLAG=1:RETURN
5210 FLAG=0:PRINT#3:PRINT#3,"OUT OF TIME...":GOTO
9800
5999 GOTO 1000
6000 REM **ODD-ONE-OUT**
6005 FOR I=1 TO 20:LET Z$=INKEY$:NEXT I
6010 CLS#3
6020 PRINT#3:PRINT#3,"            ODD-ONE-OUT"
6030 PRINT#3:PRINT#3,"WHICH OF THE FOLLOWING IS TH
E          ODD-ONE-OUT ?
6040 N1=INT(RND*(V-1)):IF M$(N1)="*" THEN GOTO 604
0
6050 N2=INT(RND*(V-1)):IF N2=N1 THEN GOTO 6050
6060 IF M$(N2)="*" THEN GOTO 6050
6070 IF M$(N2)=M$(N1) THEN N3=INT(RND*(V-1)):IF M$
(N3)=M$(N2) OR M$(N3)="*" THEN GOTO 6070
6080 IF M$(N2)<>M$(N1) THEN N3=INT(RND*(V-1)):IF M
$(N3)<>M$(N1) AND M$(N3)<>M$(N2) OR M$(N3)="*" THE
N GOTO 6080
6085 IF N1=N2 OR N2=N3 OR N1=N3 THEN GOTO 6040
6090 IF M$(N3)=M$(N1) THEN OT=N2
6100 IF M$(N2)=M$(N1) THEN OT=N3
6110 IF M$(N3)=M$(N2) THEN OT=N1
6120 PRINT#3:PRINT#3,N$(N1),N$(N2),N$(N3)
6125 PRINT#3
6130 PRINT#3:INPUT#3,"WHICH IS IT ";ANSWER$
6140 IF ANSWER$=N$(OT) THEN PRINT#3,"WHAT AN ANALY
TICAL DECISION!!!":GOTO 9700
6150 PRINT#3,"SORRY, IT IS ";N$(OT):GOTO 9800
6999 GOTO 1000
```

```
7000 REM **CAPITALS**
7005 FOR I=1 TO 20:LET Z$=INKEY$:NEXT I
7010 CLS#3
7020 PRINT#3:PRINT#3,"      THE MANIC GEOGRAPHY TE
ACHER"
7030 PRINT#3:
7040 OT=INT(RND*(V-1)):IF LEFT$(M$(OT),7)<>"CAPITA
L" THEN GOTO 7040
7050 PRINT#3,"WHAT IS THE ";M$(OT);" ?"
7060 PRINT#3:INPUT#3,ANSWER$
7070 IF ANSWER$=N$(OT) THEN PRINT#3:PRINT#3,"WOW!!
 YOU ARE CORRECT!!":GOTO 9700
7080 PRINT#3:PRINT#3,"NO, IT IS ";N$(OT):GOTO 9800
7499 GOTO 1000
7500 REM **FLASHER**
7510 CLS#3:PRINT#3:PRINT#3,"        THE FLASHER ST
RIKES!!"
7520 PRINT#3:PRINT#3,"AN ANIMAL'S NAME WILL FLASH
BEFORE YOU."
7530 LET OT=INT(RND*(V-1)):IF M$(OT)<>"AN" THEN GO
TO 7530
7540 XW=INT(RND*(41-LEN(N$(OT))))+1
7550 YW=INT(5*RND)+6
7560 LOCATE#3,XW,YW
7570 FOR I=1 TO INT(RND*5000):NEXT I
7580 PRINT#3,N$(OT)
7590 FOR I=1 TO 20:NEXT I
7600 LOCATE#3,1,YW:PRINT#3,"
                "
7610 LOCATE#3,1,6:INPUT#3,"WHAT IS THE ANIMAL ";AN
SWER$
7620 IF ANSWER$=N$(OT) THEN PRINT#3:PRINT#3,"VERY
GOOD!!!":GOTO 9700
7630 PRINT#3:PRINT#3,"NO, IT IS ";N$(OT):GOTO 9800
7999 GOTO 1000
8000 REM **THE ASTERISK**
8003 FOR I=1 TO 20:LET Z$=INKEY$:NEXT I
8005 FOR I=1 TO 20:LET Z$=INKEY$:NEXT I
8010 CLS#3:PRINT#3:PRINT#3,"          THE ASTERIS
K"
8020 PRINT#3:PRINT#3,"GUESS THE LETTERS IN THE WOR
D TO STOP   THE ASTERISK."
8030 OT=INT(RND*(V-1)):IF M$(OT)="*" THEN GOTO 803
0
8045 FOR I=1 TO LEN(N$(OT)):WORD$(I)="_":NEXT I
8046 PRINT#3:PRINT#3," ";:FOR I=1 TO LEN(N$(OT)):P
RINT#3,WORD$(I);" ";:NEXT I
8050 YW=9
8060 LOCATE#3,25,YW:PRINT#3,CHR$(249)
8070 XW=15
```

```
8090 LOCATE#3,XW,YW:PRINT#3,"*"
8100 PRINT#3:PRINT#3,"LETTER          ":LOCATE#3,8,1
1:INPUT#3,LETTER$
8110 LETTER$=LEFT$(LETTER$,1)
8120 FOR K=1 TO LEN(N$(OT))
8130 IF LETTER$=MID$(N$(OT),K,1) THEN LET WORD$(K)
=LETTER$:LOCATE#3,K*2,7:PRINT#3,LETTER$:LET FLAG=1
8135 NEXT K:IF FLAG=1 THEN FLAG=0:GOTO 8150
8140 LOCATE#3,XW,YW:PRINT#3," ":XW=XW+1
8150 FOR I=1 TO LEN(N$(OT)):IF WORD$(I)="_" THEN 8
160
8155 NEXT I:GOTO 8300
8160 IF XW<25 THEN GOTO 8090
8170 LOCATE#3,XW,YW:PRINT#3,"*":PRINT#3:PRINT#3,"Y
OU´VE BEEN SQUASHED!!!":GOTO 9800
8300 PRINT#3:PRINT#3:PRINT#3,"YOU HAVE SURVIVED!!!
":GOTO 9700
8500 REM **NO HAZARD**
8510 CLS#3
8520 LOCATE#3,10,6:PRINT#3,"WOT..NO HAZARD"
8530 GOTO 9700
9000 REM ***MOVEMENT***
9001 LET X1=X:LET Y1=Y
9010 IF INKEY(0)<>0 THEN GOTO 9025
9015 LET Y1=Y-1
9020 GOTO 9100
9025 IF INKEY(2)<>0 THEN GOTO 9035
9030 LET Y1=Y+1
9032 GOTO 9100
9035 IF INKEY(8)<>0 THEN GOTO 9060
9040 IF Y=7 AND X=2 THEN GOTO 9010
9045 LET X1=X-1
9050 GOTO 9100
9060 IF INKEY(1)<>0 THEN GOTO 9010
9065 IF Y=8 AND X=39 THEN GOTO 9900
9070 LET X1=X+1
9100 IF A$(Y1,X1)="*" THEN LET DEAD=1 ELSE LET DEA
D=0
9110 LOCATE#2,X,Y:PRINT#2," ":
9115 LET X=X1:LET Y=Y1
9120 LOCATE#2,X,Y:IF DEAD=0 THEN IF X/2=INT(X/2)TH
EN PRINT#2,CHR$(250) ELSE PRINT#2,CHR$(251)
9130 IF DEAD=0 THEN RETURN
9200 CLS#3
9210 LOCATE#3,8,6:PRINT#3,"YOU STRAYED OFF THE PAT
H!!"
9215 EI
9220 FOR I=1 TO 2000:NEXT I:GOTO 9800
9230 GOTO 9230
9600 REM **LOAD DATABASE**
```

```
9601 MODE 1:BORDER 1:INK 0,1:INK 1,24:PAPER 0:PEN
1:CLS:LOCATE 1,12:PRINT"PUT DATA CASSETTE INTO DRI
VE & HIT ENTER"
9602 IF INKEY$="" THEN GOTO 9602
9610 V=1
9615 DIM N$(1001),M$(1001)
9620 OPENIN "!EDBASE"
9630 INPUT#9,N$(V)
9640 IF N$(V)="XXX" THEN GOTO 9680
9650 INPUT#9,M$(V):IF M$(V)="*" THEN GOTO 9630
9660 V=V+1
9670 GOTO 9630
9680 CLOSEIN
9690 RETURN
9700 REM **CLEAR MESSAGE WINDOW**
9701 FOR I=1 TO 2000:NEXT I
9710 CLS#3
9720 LOCATE #3,1,6:PRINT#3,"AND NOW TO CONTINUE YO
UR JOURNEY..."
9730 GOTO 1000
9800 REM **DEAD ROUTINE**
9801 FOR I=1 TO 2000:NEXT I
9810 INK 0,6
9830 CLS#3
9840 LOCATE#3,15,6:PRINT#3,"YOU ARE DEAD."
9841 SOUND 1,213,100:SOUND 1,159,250
9842 SOUND 1,213,100:SOUND 1,127,250:SOUND 1,127,5
0:SOUND 1,106,50:SOUND 1,142,200
9843 SOUND 1,127,50:SOUND 1,159,150:SOUND 1,142,50
:SOUND 1,190,150:SOUND 1,169,50:SOUND 1,213,200
9844 SOUND 1,159,250:SOUND 1,213,100:SOUND 1,127,2
50:SOUND 1,127,50:SOUND 1,106,50:SOUND 1,142,150
9845 SOUND 1,127,50:SOUND 1,159,150:SOUND 1,142,50
:SOUND 1,190,50:SOUND 1,169,50:SOUND 1,213,50:SOUN
D 1,159,150
9849 FOR I=1 TO 20:LET Q$=INKEY$:NEXT I
9850 LOCATE #3,8,10:PRINT#3,"HIT ENTER FOR ANOTHER
 GO..."
9860 LET Q$=INKEY$:IF Q$=""THEN GOTO 9860
9870 GOTO 10
9900 REM **END ROUTINE**
9910 MODE 0
9920 WINDOW#1,1,20,12,40
9930 LOCATE#1,7,8:PRINT#1,"SUCCESS!!"
9940 WINDOW#2,1,20,1,11
9950 FOR I=1 TO 500:PAPER#2,5:CLS#2:INK 5,INT(RND*
26)+1:SOUND 1,INT(RND*1000),10
9960 NEXT I
```

9. The Card Player

This game is designed to develop your ability to select a word from a given list of letters and thereby develop correct spelling and an extended vocabulary. The game is for two to four players with the computer playing any of these hands. The game can also be played in two different modes:

1. An ordinary game in which no checks are made against you cheating.
2. Extending the database – where the computer will check each word you enter and if it is unable to find the word in its memory, you are asked to extend the database by giving its meaning.

Instructions

The game consists of several rounds and ends when a player reaches one hundred points. Seven cards are dealt to each player with the remainder of the pack being placed in a pile, face down on the table. One of the players commences the round and then each player takes a turn in sequence. At his turn, each player must examine his cards and either

1. form a complete word and place it face up on the table, or
2. 'pass' his turn and accept another card from the unexposed pack unless there are no cards left or he already has thirteen cards in his hand.

After placing a word, if there are enough cards remaining in the unseen pack the player should add cards to his hand to make the total number up to seven. If he already has more then seven then no action is taken at this point.

When there are no cards left in the unexposed pack, the object of the game is to dispose of as many cards as possible from your hand. The round ends when one player has placed all his cards on

the table or when each player has passed his turn in a complete circuit of the table.

Scoring

Each letter has an associated value, and each time a word is placed on the table its value is calculated and added to the player's score. If at any time a score reaches one hundred or more then the game is over. At the end of each round and at the end of the game the total value of each player's hand is calculated and then deducted from his score.

The cards

There are fifty-two cards in the pack with a varying number of cards for each letter. Each letter also has its own value and these are summarised in the table below.

Letter	No.	Value	Letter	No.	Value
A	4	2	N	1	4
B	1	6	O	3	2
C	1	6	P	1	4
D	1	4	Q	1	10
E	4	2	R	3	4
F	1	6	S	3	2
G	1	6	T	3	4
H	3	4	U	4	2
I	4	2	V	1	4
J	1	10	W	3	4
K	1	6	X	1	8
L	3	4	Y	1	6
M	1	4	Z	1	10

The rules of the game must remain the same, but there is no reason why you should not experiment with a different pack of fifty-two cards or with the letters having different values. The winning number of points can be also altered. All this is explained later in this chapter.

The program

When the program is executed you are asked to load the database. The amount of time this takes depends on the number of words which are stored in the file, but the process can take about fifteen minutes. When the Amstrad has finished reorganising its memory the screen will show several windows and you will be asked which style of game you require, as explained above. When you have selected it, you are asked for the number of players and their names. The computer can play the hand of as many players as you require and you must tell the machine which it will be expected to play. In one of the windows is displayed the scores of the players and in another the words they have created are shown. The remaining two windows are used to show the hand of the player whose turn it is and useful information such as the number of cards that are left.

The pack is shuffled, the cards are dealt and it is the first player's turn. If you wish to know the value of each of the cards then this is displayed if the key '2' is entered. You can also see the number of each available letter – to do this the key '3' should be entered. After either of these functions the key '0' will return you back to the main menu. You can then enter your word. If you cannot find a word, pressing the key '1' will pass the play to the next player.

If you are playing a style 1 game, in which no checks are performed, after you have input your word you are asked if you wish to accept it, and if you do the word is displayed by your name and you are given the appropriate number of points. If you reject the word then you must try again. If, however, you are playing a style 2 game, before this happens the computer will check its memory to see if it can find your word. If it can then play continues in the way described above, otherwise you are asked the meaning of the word and it is added to the database.

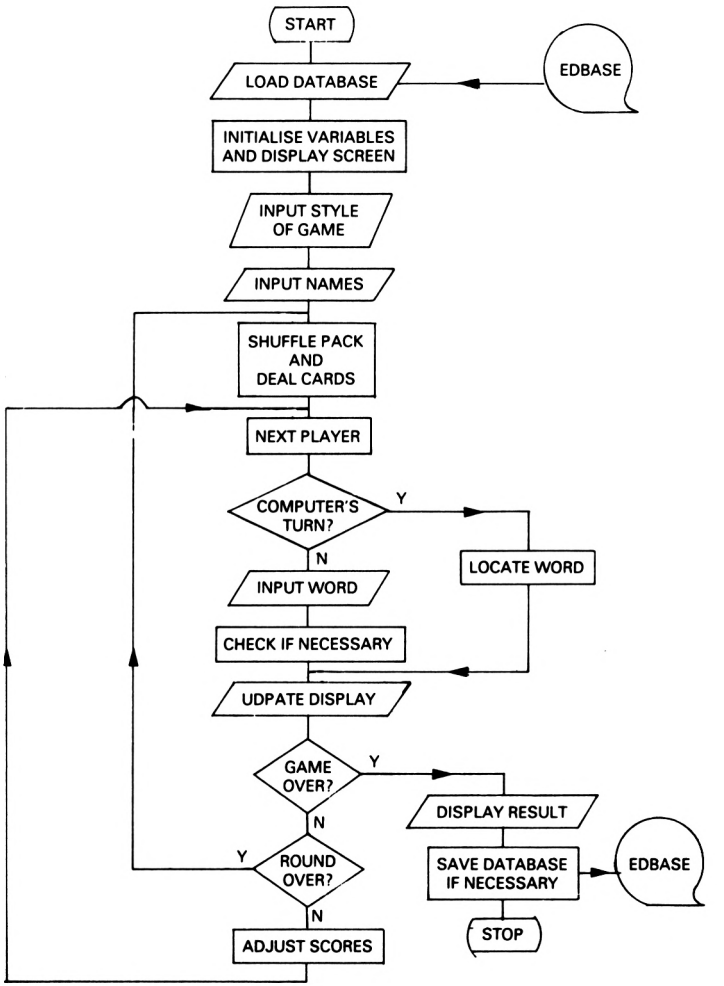
When one of the players has laid all his cards or all the players have passed and no further play can take place, the round is over and the scores are adjusted before the pack is shuffled and redealt. When a player reaches one hundred points the game is over and that player is declared the winner.

When the computer is playing it takes about three minutes to decide on its word when there are a thousand words in the database. Sometimes it uses up all its memory and has to perform

a garbage collect to find extra space. This slows the game down considerably, but gives you a chance to play the computer's turn and spot any words which are missing from the database.

Flowchart

The following flowchart shows the 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.

WN	Winning number
GAO	Set to one if the game is over
PASS	Counts the number of consecutive passes
HAND()	The cards currently held in the hands
PACK()	The order of the cards in the shuffled pack
NHAND()	The number of cards in each hand
NCARD	The next card in the pack
NP$()	Players' names
CP$()	Shows if the computer is playing a given hand
CARD$()	The pack of cards
VALUE()	The value of each card
SCORE	The score of the current word
LEV$	The style of game
SCORE	Players' scores
WD$	The chosen word
NP$	The character representing the number of players
NP	The numerical equivalent of NP$
WORD$()	The words from the database
CLUE$()	The clues from the database
CS$()	The categories
NW	The number of words
NC	The number of categories

Comments on the program

The program is structured, and the following notes show how the individual routines operate.

Lines 10-70
This section of program calls several of the later routines to create the initial display and to give the variables their values. In line 30 the routine to read in the database is called, and in line 20 WN is given its value. If you think that one hundred is too many points to aim for or if you feel that it is not enough, by altering this line you

can change the score required to win.

Lines 80-580
This is the main section of the program which handles the management of the subroutines used throughout the game. It is here that the players input their words and that the scores and screen are adjusted accordingly.

Lines 590-810
The routine used when the computer has to select the word from the database.

Lines 820-880
A routine to assess the position when a player passes. If the player has less than thirteen cards and there are some remaining in the unexposed pack then he is given another card.

Lines 890-980
This routine adds a sufficient number of cards to give a player seven after a word has been laid.

Lines 990-1060
This section of code is used when there are no cards left in the pack.

Lines 1070-1230
When the round is over the scores are adjusted and another round begins.

Lines 1240-1320
This is the end game routine which offers another match when one has been completed.

Lines 1330-1410
This displays the instructions in one of the windows.

Lines 1420-1650
This routine displays the values of the cards or the number of each card when requested.

Lines 1660-1810
This part of the program is used if a style 2 game is in operation. It interrogates the database to see if a player's word is present and if it cannot be found then information about its meaning is sought

and added to the file.

Lines 1820-2000
This routine rewrites the database if any new words have been added.

Lines 2010-2220
This initial routine dimensions the arrays, defines the colours and creates the windows.

Lines 2230-2310
This is where the style of game to be played is selected.

Lines 2320-2640
Here the players' names are input and windows for displaying used words are redefined if necessary. It is also within these lines that you must tell the computer if it is playing any of the hands.

Lines 2650-2720
In these lines the pack is defined. To redefine the pack to one of your own choosing, the lines 2700 and 2710 must be altered. There must be fifty-two cards and the number following each card represents its value. For example,

```
2700 DATA A,2,A,2,A,2,B,4,B,4
```

will create three cards with the letter A, each worth 2 points, and two cards with the letter B, each worth 4 points.

Lines 2730-2840
This routine shuffles the pack.

Lines 2850-3060
This subroutine reads in the database.

The listing

The listing should be entered very carefully and then saved on to cassette at 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
20 LET WN=100
30 GOSUB 2850
40 GOSUB 2010
50 GOSUB 2230
60 GOSUB 2320
70 GOSUB 2650
80 REM ** PLAY GAME **
90 LET GAO=0:LET WP=0:LET PASS=0
100 GOSUB 2730
110 FOR I=1 TO NP
120 FOR J=1 TO 7
130 LET HAND(I,J)=PACK(J+7*(I-1))
140 NEXT J
150 FOR K=8 TO 13:LET HAND(I,K)=0:NEXT K
160 LET NHAND(I)=7
170 NEXT I
180 LET NCARD=7*NP+1
190 LET I=ST
200 IF I=NP THEN LET I=0
210 LET I=I+1
220 GOSUB 1330
230 CLS #3:PLOT 0,0,1:DRAWR 0,96:DRAWR 464,0:DRAWR
 0,-96:DRAWR -464,0
240 LOCATE #3,3,2:PRINT #3,NP$(I)
250 PEN #3,3
260 LOCATE #3,3,3
270 FOR J=1 TO 14
280 IF HAND(I,J)<>0 THEN PRINT #3,CARD$(HAND(I,J))
;" ";
290 NEXT J
300 PEN #3,1
310 LOCATE #3,3,4:PRINT #3,"YOUR WORD  ";
320 LOCATE #3,14,4:PRINT #3,"          "
330 LOCATE #3,14,4
340 IF CP$(I)="Y" THEN GOTO 590
350 LET SCORE=0:LET WD$="":LET Z$=""
360 FOR K=1 TO 14
370 LET TST(K)=HAND(I,K)
380 NEXT K
390 WHILE Z$<>CHR$(13)
400 LET Z$=INKEY$:IF Z$="" THEN GOTO 400
410 IF Z$="2" OR Z$="3" THEN GOSUB 1420
420 IF Z$="1" THEN LET PASS=PASS+1:GOTO 820
430 FOR K=1 TO 14
440 IF CARD$(TST(K))=Z$ THEN LET SCORE=SCORE+VALUE
(TST(K)):LET TST(K)=0:PRINT #3,Z$;:LET WD$=WD$+Z$:
GOTO 400
450 NEXT K
460 LET PASS=0
```

```
470 WEND
480 IF LEV$="2" THEN GOSUB 1660
490 LOCATE #3,10,5:PRINT #3,"ACCEPT (Y/N)"
500 LET Q$=INKEY$:IF Q$="" THEN GOTO 500
510 IF Q$<>"Y" THEN LOCATE #3,10,5:PRINT#3,"
      ";:GOTO 230
520 LET SCORE(I)=SCORE(I)+SCORE:IF LEN(WD$)>6 THEN
 LET SCORE(I)=SCORE(I)+30
530 PRINT #(I+3),WD$
540 LOCATE #1,7,I+3:PRINT #1,USING "###";SCORE(I)
550 GOSUB 890
560 IF NHAND(I)=0 THEN GOTO 1070
570 IF SCORE(I)>=WN THEN LET GAO=1:LET WP=I:GOTO 1
120
580 GOTO 200
590 REM ** COMPUTER TO PLAY **
600 LET MX=0
610 FOR J=1 TO NW
620 FOR F=1 TO 13
630 LET TST(F)=HAND(I,F)
640 NEXT F
650 LET SCORE=0
660 FOR L=1 TO LEN(WORD$(J))
670 IF FRE(0)<2048 THEN LET D=FRE("")
680 FOR K=1 TO NHAND(I)
690 IF CARD$(TST(K))=MID$(WORD$(J),L,1) THEN LET S
CORE=SCORE+VALUE(TST(K)):LET TST(K)=0:GOTO 720
700 NEXT K
710 GOTO 740
720 NEXT L
730 IF SCORE>MX THEN LET MX=SCORE:LET WORD$=WORD$(
J)
740 NEXT J
750 IF MX=0 THEN PRINT #3,"PASS":FOR Z=1 TO 1000:N
EXT Z:LET Z$="1":GOTO 420
760 LET WD$=WORD$:LET SCORE=MX
770 PRINT #3,WD$
780 FOR Z=1 TO 1000:NEXT Z
790 GOTO 520
800 IF INKEY$="" THEN GOTO 800
810 MODE 1:END
820 REM ** PLAYER PASSES **
830 IF NCARD=53 AND PASS=NP THEN GOTO 1070
840 IF NCARD=53 OR NHAND(I)=13 THEN GOTO 580
850 LET NHAND(I)=NHAND(I)+1
860 LET HAND(I,NHAND(I))=PACK(NCARD)
870 LET NCARD=NCARD+1
880 GOTO 580
890 REM ** REPLACE LETTERS **
900 LET NHAND(I)=NHAND(I)-LEN(WD$)
```

```
910 FOR K=1 TO LEN(WD$)
920 IF NCARD=53 OR NHAND(I)>6 THEN GOTO 990
930 LET Z$=MID$(WD$,K,1)
940 FOR L=1 TO 14
950 IF CARD$(HAND(I,L))=Z$ THEN LET HAND(I,L)=PACK
(NCARD):LET NCARD=NCARD+1:NHAND(I)=NHAND(I)+1:GOTO
 970
960 NEXT L
970 NEXT K
980 RETURN
990 REM ** NO CARDS LEFT **
1000 FOR M=K TO LEN(WD$)
1010 LET Z$=MID$(WD$,M,1)
1020 FOR L=1 TO 14
1030 IF CARD$(HAND(I,L))=Z$ THEN LET HAND(I,L)=0:G
OTO 1050
1040 NEXT L
1050 NEXT M
1060 RETURN
1070 REM ** END OF ROUND **
1080 LET ST=I
1090 CLS #3:PLOT 0,0,1:DRAWR 0,96:DRAWR 464,0:DRAW
R 0,-96:DRAWR -464,0
1100 LOCATE #3,9,2:PRINT #3,"ROUND OVER"
1110 LOCATE #3,5,4:PRINT #3,"ADJUSTING THE SCORES"
1120 FOR I=1 TO NP
1130 IF I=WP THEN CLS #(I+3):GOTO 1200
1140 FOR J=1 TO 13
1150 IF HAND(I,J)<>0 THEN LET SCORE(I)=SCORE(I)-VA
LUE(HAND(I,J))
1160 NEXT J
1170 IF SCORE(I)<0 THEN LET SCORE(I)=0
1180 CLS #(I+3)
1190 LOCATE #1,7,I+3:PRINT #1,USING "###";SCORE(I)
1200 NEXT I
1210 FOR Z=1 TO 1000:NEXT Z
1220 IF GAO=1 THEN GOTO 1240
1230 LET I=ST:GOTO 80
1240 REM ** GAME OVER **
1250 CLS #3:PLOT 0,0,1:DRAWR 0,96:DRAWR 464,0:DRAW
R 0,-96:DRAWR -464,0
1260 LOCATE #3,10,2:PRINT #3,"GAME OVER"
1270 LOCATE #3,6,4:PRINT #3,"ANOTHER GO (Y/N)?"
1280 LET AG$=INKEY$:IF AG$="" THEN GOTO 1280
1290 IF AG$="Y" THEN GOSUB 2030:GOSUB 2230:GOSUB 2
320:FOR I=1 TO 4:LET SCORE(I)=0:NEXT I:GOTO 80
1300 IF EXT=1 THEN GOSUB 1820
1310 IF INKEY$="" THEN GOTO 1310
1320 END
1330 REM ** INSTRUCTION WINDOW **
```

```
1340 LOCATE #2,2,2:PRINT #2,"CARDS":LOCATE #2,2,3:
PRINT #2,"LEFT":LOCATE #2,7,3:PEN #2,3:PRINT #2,US
ING "##";53-NCARD
1350 PEN #2,1:LOCATE #2,2,5:PRINT #2,"PRESS..."
1360 LOCATE #2,2,7:PRINT #2,"1. PASS"
1370 LOCATE #2,2,9:PRINT #2,"2. VALUE"
1380 LOCATE #2,2,11:PRINT #2,"3. NO."
1390 LOCATE #2,2,14:PRINT #2,"0.  TO"
1400 LOCATE #2,2,15:PRINT #2,"  RETURN"
1410 RETURN
1420 REM ** ASSISTANCE **
1430 CLS #2:PLOT 478,254,1:DRAWR 160,0:DRAWR 0,-25
4:DRAWR -160,0:DRAWR 0,254
1440 LET H=0
1450 IF Z$="3" THEN GOTO 1550
1460 FOR F=1 TO 52
1470 LET C$=CARD$(F)
1480 IF C$<>C1$ AND H<13 THEN LET C1$=C$:LET H=H+1
:LOCATE #2,2,H+2:PEN #2,3:PRINT #2,CARD$(F);:PEN #
2,1:PRINT #2,USING "##";VALUE(F)
1490 IF C$<>C1$ AND H>12 THEN LET C1$=C$:LET H=H+1
:LOCATE #2,7,H-11:PEN #2,3:PRINT #2,CARD$(F);:PEN
#2,1:PRINT #2,USING "##";VALUE(F)
1500 NEXT F
1510 LET S$=INKEY$:IF S$<>"0" THEN GOTO 1510
1520 CLS #2:PLOT 478,254,1:DRAWR 160,0:DRAWR 0,-25
4:DRAWR -160,0:DRAWR 0,254
1530 GOSUB 1330
1540 RETURN
1550 LET F=1
1560 FOR H=1 TO 26
1570 LET C$=CHR$(64+H)
1580 IF H<14 THEN LOCATE #2,2,H+2:PEN #2,3:PRINT #
2,C$;
1590 IF H>13 THEN LOCATE #2,7,H-11:PEN #2,3:PRINT
#2,C$;
1600 PEN #2,1
1610 LET T=0
1620 IF C$=CARD$(F) THEN LET T=T+1:LET F=F+1:IF C$
<>"Z" THEN GOTO 1620
1630 PRINT #2,USING "##";T
1640 NEXT H
1650 GOTO 1510
1660 REM ** CHECK PLAYER´S WORD **
1670 FOR Z=1 TO NW
1680 IF WORD$(Z)=WD$ THEN RETURN
1690 NEXT Z
1700 CLS #3:PLOT 0,0,1:DRAWR 0,96:DRAWR 464,0:DRAW
R 0,-96:DRAWR -464,0
1710 LOCATE #3,2,2:PRINT #3,"I DO NOT KNOW ";WD$
```

```
1720 LOCATE #3,2,3:PRINT #3,"DO YOU KNOW ITS MEANI
NG"
1730 LET S$=INKEY$:IF S$="" THEN GOTO 1730
1740 IF S$="N" THEN RETURN
1750 IF S$<>"Y" THEN GOTO 1730
1760 LOCATE #3,2,3:PRINT #3,"WHAT IS THE MEANING?
  "
1770 LOCATE #3,2,4:PEN #3,3
1780 LINE INPUT #3,CLUE$(NW+1)
1790 LET WORD$(NW+1)=WD$:LET NW=NW+1
1800 LET EXT=1
1810 PEN #3,1:RETURN
1820 REM ** REWRITE DATABASE **
1830 CLS:PLOT 0,110,1:DRAWR 0,288:DRAWR 464,0:DRAW
R 0,-288:DRAWR -464,0
1840 WINDOW #0,2,28,2,17
1850 LOCATE 8,2:PRINT"RESAVE DATABASE"
1860 PRINT:PRINT:PRINT
1870 PRINT"PLACE A BLANK CASSETTE INTOTHE DATACORD
ER"
1880 PRINT:PRINT:PRINT
1890 OPENOUT "WORDS"
1900 FOR I=1 TO NW
1910 PRINT #9,WORD$(I)
1920 PRINT #9,CLUE$(I)
1930 NEXT I
1940 PRINT #9,"XXX"
1950 FOR I=1 TO NC
1960 PRINT #9,CS$(I,1)
1970 PRINT #9,CS$(I,2)
1980 NEXT I
1990 CLOSEOUT
2000 RETURN
2010 REM ** INITIAL ROUTINE **
2020 DIM PS(4),PN$(4),CP$(4),HAND(4,20),NHAND(4),T
ST(20),SCORE(4)
2030 MODE 1
2040 INK 0,26:INK 1,0:INK 2,18:INK 3,6
2050 PAPER 2:CLS:BORDER 18
2060 WINDOW #1,31,40,1,8:PAPER #1,0:PEN #1,1:CLS #
1
2070 PLOT 478,398,1:DRAWR 160,0:DRAWR 0,-126:DRAWR
 -160,0:DRAWR 0,126
2080 PRINT #1:PRINT #1,"  SCORES"
2090 PEN #1,3
2100 WINDOW #2,31,40,10,25:PAPER #2,0:CLS #2
2110 PLOT 478,254,1:DRAWR 160,0:DRAWR 0,-254:DRAWR
 -160,0:DRAWR 0,254
2120 WINDOW #3,1,29,20,25:PAPER #3,0:PEN #3,1:CLS
#3
```

```
2130 PLOT 0,0,1:DRAWR 0,96:DRAWR 464,0:DRAWR 0,-96
:DRAWR -464,0
2140 WINDOW #0,1,29,1,18:PAPER #0,0:PEN #0,1:CLS #
0
2150 PLOT 0,110,1:DRAWR 0,288:DRAWR 464,0:DRAWR 0,
-288:DRAWR -464,0
2160 LOCATE #0,10,2:PRINT #0,"WORDS PLAYED "
2170 WINDOW #4,2,14,5,10:PAPER #4,0:PEN #4,1:CLS #
4
2180 WINDOW #5,16,28,5,10:PAPER #5,0:PEN #5,1:CLS
#5
2190 WINDOW #6,2,14,13,18:PAPER #6,0:PEN #6,1:CLS
#6
2200 WINDOW #7,16,28,13,18:PAPER #7,0:PEN #7,1:CLS
 #7
2210 PEN #0,3
2220 RETURN
2230 REM ** CHOOSE LEVEL **
2240 LOCATE #3,4,2:PRINT #3,"WHICH STYLE OF GAME ?
"
2250 LOCATE #3,4,4:PRINT #3,"1. ORDINARY GAME"
2260 LOCATE #3,4,5:PRINT #3,"2. EXTENDING DATABASE
"
2270 LET LEV$=INKEY$
2280 IF LEV$="1" OR LEV$="2" THEN GOTO 2300
2290 GOTO 2270
2300 CLS #3:PLOT 0,0,1:DRAWR 0,96:DRAWR 464,0:DRAW
R 0,-96:DRAWR -464,0
2310 RETURN
2320 REM ** INPUT NAMES **
2330 CLS #3:PLOT 0,0,1:DRAWR 0,96:DRAWR 464,0:DRAW
R 0,-96:DRAWR -464,0
2340 LOCATE #3,3,3:PRINT #3,"HOW MANY PLAYERS (2-4
)? ";
2350 LET NP$=INKEY$:IF NP$<"2" OR NP$>"4" THEN GOT
O 2350
2360 PRINT #3,NP$
2370 LET NP=VAL(NP$)
2380 IF NP=2 THEN WINDOW #4,2,14,5,17:WINDOW #5,16
,28,5,17:PAPER #4,0:PAPER #5,0:PEN #4,1:PEN #5,1:C
LS #4:CLS #5
2390 IF NP=3 THEN WINDOW #6,9,21,13,18:PAPER #6,0:
PEN #7,1:CLS #6
2400 FOR Z=1 TO 1000:NEXT Z
2410 CLS #3:PLOT 0,0,1:DRAWR 0,96:DRAWR 464,0:DRAW
R 0,-96:DRAWR -464,0
2420 LOCATE #3,9,2:PRINT #3,"ENTER  NAMES"
2430 FOR I=1 TO NP
2440 LOCATE #3,2,3:PRINT #3,SPACE$(26)
2450 LOCATE #3,2,4:PRINT #3,SPACE$(26)
```

```
2460 LOCATE #3,2,5:PRINT #3,SPACE$(26)
2470 LOCATE #3,3,4:PRINT #3,"PLAYER";I;"  ";:LINE
INPUT #3,NP$(I)
2480 LET NP$(I)=LEFT$(NP$(I),4)
2490 IF LEN(NP$(I))<4 THEN LET NP$(I)=NP$(I)+SPACE
$(4-LEN(NP$(I)))
2500 LOCATE #3,3,5:PRINT #3,"CAN AMSTRAD PLAY FOR
";NP$(I);
2510 LET CP$(I)=INKEY$
2520 IF CP$(I)="Y" OR CP$(I)="N" THEN GOTO 2540
2530 GOTO 2510
2540 LOCATE #1,1,3+I
2550 IF CP$(I)="Y" THEN PRINT #1,CHR$(144);NP$(I);
"   0";
2560 IF CP$(I)="N" THEN PRINT #1,CHR$(32);NP$(I);"
   0";
2570 IF I=1 THEN LOCATE 2,4:PRINT NP$(I)
2580 IF I=2 THEN LOCATE 16,4:PRINT NP$(I)
2590 IF I=3 AND NP=3 THEN LOCATE 9,12:PRINT NP$(I)
2600 IF I=3 AND NP=4 THEN LOCATE 2,12:PRINT NP$(I)
2610 IF I=4 THEN LOCATE 16,12:PRINT NP$(I)
2620 NEXT I
2630 CLS #3:PLOT 0,0,1:DRAWR 0,96:DRAWR 464,0:DRAW
R 0,-96:DRAWR -464,0
2640 RETURN
2650 REM ** DEFINE CARDS **
2660 DIM CARD$(52),VALUE(52),PACK(52)
2670 FOR I=1 TO 52
2680 READ CARD$(I),VALUE(I)
2690 NEXT I
2700 DATA A,2,A,2,A,2,A,2,B,6,C,6,D,4,E,2,E,2,E,2,
E,2,F,6,G,6,H,4,H,4,H,4,I,2,I,2,I,2,I,2,J,10,K,6,L
,4,L,4,L,4
2710 DATA M,4,N,4,O,2,O,2,O,2,P,4,Q,10,R,4,R,4,R,4
,S,2,S,2,S,2,T,4,T,4,T,4,U,2,U,2,U,2,U,2,V,4,W,4,W
,4,W,4,X,8,Y,6,Z,10
2720 RETURN
2730 REM ** SHUFFLE PACK **
2740 CLS #3:PLOT 0,0,1:DRAWR 0,96:DRAWR 464,0:DRAW
R 0,-96:DRAWR -464,0
2750 LOCATE #3,9,3:PRINT #3,"DEALING CARDS"
2760 FOR I=1 TO 52
2770 LET R=1+INT(RND(1)*52)
2780 FOR J=1 TO I
2790 IF PACK(J)=R THEN GOTO 2770
2800 NEXT J
2810 LET PACK(I)=R
2820 NEXT I
2830 CLS #3:PLOT 0,0,1:DRAWR 0,96:DRAWR 464,0:DRAW
R 0,-96:DRAWR -464,0
```

```
2840 RETURN
2850 REM ** READ IN FILE **
2860 DIM WORD$(1001),CLUE$(1001),CS$(40,2)
2870 CLS
2880 LOCATE 14,2:PRINT"THE CARD GAME"
2890 PRINT:PRINT:PRINT
2900 PRINT" PLACE THE DATA CASSETTE INTO THE
  DATACORDER."
2910 PRINT:PRINT:PRINT:PRINT
2920 OPENIN "!EDBASE"
2930 LET NW=1
2940 INPUT #9,WORD$(NW)
2950 IF WORD$(NW)="XXX" THEN LET NW=NW-1:GOTO 2990
2960 INPUT #9,CLUE$(NW)
2970 LET NW=NW+1
2980 GOTO 2940
2990 LET NC=1
3000 IF EOF THEN LET NC=NC-1:GOTO 3050
3010 INPUT #9,CS$(NC,1)
3020 INPUT #9,CS$(NC,2)
3030 LET NC=NC+1
3040 GOTO 3000
3050 CLOSEIN
3060 RETURN
```

10. Hangman

This game, like many of the others in the book, wil! improve if the database is extended to a larger size, for it chooses the words which you must guess from this data. This is the standard version of Hangman in which you are given a limited number of guesses at the letters in a word, and for each incorrect guess a part of a gallows containing a sentenced man will appear. Should the whole man be displayed on the screen then you have been hanged, and naturally you have lost!

Instructions

Hangman is really a game for one person against the computer, but it can be extended to a group activity by several people taking turns to guess a letter or each person having his own word and comparing the number of guesses needed to obtain the solution.

When the program is executed, the user is asked to load the words in the database into the computer's memory. As explained previously, the length of time this takes is dependent on the number of words, and while it is happening it may appear that the computer is not functioning. When the process is over the screen shows the empty picture on the right and the title 'Hangman' on the left. The message 'selecting word' will flash while the computer examines its memory for a word at random. When it has chosen, a number of asterisks are shown, each one corresponding to a letter in the word.

To play the game, just press the key corresponding to the letter which you think is in the word. If it is then it will appear in its correct position or positions, otherwise a portion of the picture will be drawn and the letter will appear in the used letters section of the screen. If you press a key which you have already used then a message is flashed on the screen and you can input another choice.

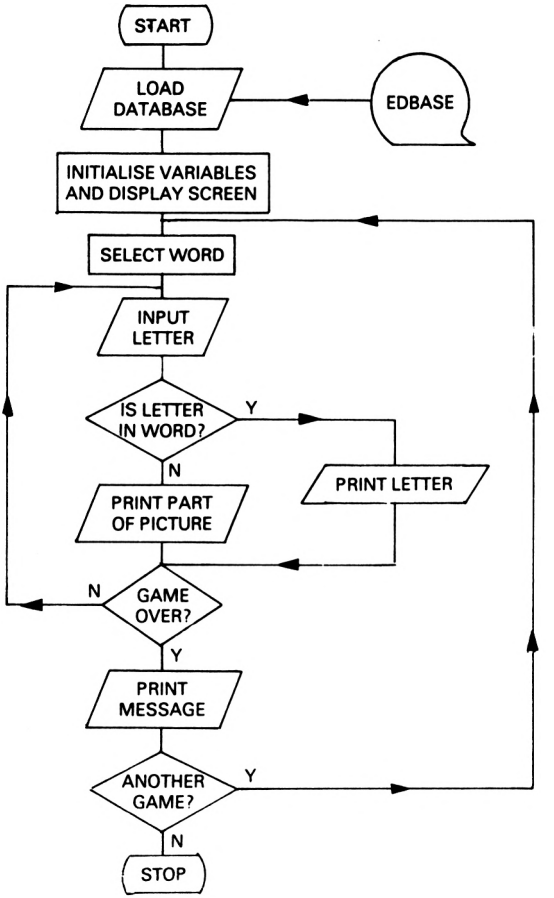
The game finishes either when you have completed the word, in which case you win, or when the computer has drawn the

gallows and man, in which case you have lost and the word is displayed on the screen. You have the option of another go or ending the program, and the computer takes action depending on your response to its prompt.

The program

Flowchart

The flowchart 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.

UL$()	Used letters
W$	Word
L$	Letter input from keyboard
UL	Number of incorrect letters used
COR	Number of correct letters
WR	Indicates next portion of picture
A$()	Words read from the database
N	The number of words
AG$	Another go variable

Comments on the program

The program is highly structured with the majority of the code being contained within subroutines. Each subroutine is clearly labelled in the listing and explained in the notes given below.

Lines 10-220
The section of program calls all the initialising subroutines and contains the main program which employs many of the later routines.

Lines 230-280
A portion of code which is used to call the appropriate drawing routine depending on the value of the variable WR.

Lines 290-420
This section of the program is concerned with the loading of the database.

Lines 430-500
Here the word is chosen at random from all of the words stored in the computer's memory in the array A$(). Each word is referenced by a number, e.g. A$(51), and the word is chosen by generating a number in the range 1 to N (number of words).

Lines 510-680
This routine creates the initial screen and defines the inks and windows to be used.

Lines 690-1480
This section of the program is used to construct the gallows and the man on the screen. It is divided into nine sections, each providing a different part of the picture.

690-720	Base of the gallows
730-780	Side of the gallows
790-800	Top of the gallows and the noose
870-990	The face
1000-1120	The body
1130-1210	The right arm
1220-1300	The left arm
1310-1390	The right leg
1400-1480	The left leg

Lines 1490-1540
This routine prints the congratulations message on the screen when the word is guessed correctly.

Lines 1550-1700
This routine is called if the man has been hanged. It also displays the correct word directly beneath your attempt!

Lines 1710-1770
This is where the computer gives you the choice of another word or ending the program.

Lines 1780-1980
The subroutine which creates the user-defined graphics which are required to draw the man.

Lines 1990-2100
This is the short routine which is concerned with redefining assorted variables if another word is required.

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 DIM UL$(26)
20 GOSUB 1780
30 GOSUB 290
40 GOSUB 510
50 GOSUB 430
60 PEN #2,2
70 LET ST=6-INT(LEN(W$)/2)
80 LOCATE #2,ST,10:PRINT #2,STRING$(LEN(W$),"*")
90 PEN #2,1
100 LET L$=INKEY$:IF L$="" THEN GOTO 100
110 IF ASC(L$)<65 OR ASC(L$)>90 THEN GOTO 100
120 FOR Z=1 TO UL
130 IF L$=UL$(Z) THEN PRINT #7," ALREADY   USED":F
OR ZZ=1 TO 2000:NEXT ZZ:CLS #7:GOTO 100
140 NEXT Z
150 LET UL$(UL)=L$:LET UL=UL+1
160 REM ** CHECK FOR LETTER **
170 LET FL=0
180 FOR I=1 TO LEN(W$)
190 IF MID$(W$,I,1)=L$ THEN LOCATE #2,I+ST-1,10:PR
INT #2,L$:LET COR=COR+1:LET FL=1
200 NEXT I
210 IF COR=LEN(W$) THEN GOSUB 1490
220 IF FL=1 THEN GOTO 100
230 REM ** WRONG LETTER **
240 LET WR=WR+1
250 LOCATE #2,WR+1,23:PRINT #2,L$
260 ON WR GOSUB 690,730,790,870,1000,1130,1220,131
0,1400
270 IF WR=9 THEN GOSUB 1550
280 GOTO 100
290 REM ** INPUT DATABASE **
300 DIM A$(1001)
310 MODE 1:INK 0,24:INK 1,6:CLS
320 PRINT:PRINT:PRINT TAB(16)"HANGMAN"
330 PRINT:PRINT:PRINT:PRINT"          ENTER THE DA
TABASE"
340 PRINT:PRINT:PRINT"PLACE DATABASE CASSETTE  INT
```

```
O DATACORDER"
350 PRINT:PRINT:PRINT
360 OPENIN "!EDBASE"
370 LET N=1
380 INPUT #9,A$(N)
390 IF EOF THEN CLOSEIN:RETURN
400 INPUT #9,A$
410 LET N=N+1
420 GOTO 380
430 REM ** SELECT WORD **
440 PRINT #7,"SELECTING  WORD"
450 FOR Z=1 TO 1000:NEXT Z
460 LET R=1+INT(N*RND(1))
470 IF LEN(A$(R))<3 OR LEN(A$(R))>9 THEN GOTO 460
480 LET W$=A$(R)
490 CLS #7
500 RETURN
510 REM ** INITIAL SCREEN **
520 MODE 0
530 INK 0,26:INK 1,6:INK 2,2:INK 3,0:INK 4,24:INK
5,11:INK 6,22:INK 7,15:INK 8,6,24:INK 11,6:INK 12,
20:INK 13,8:INK 14,1
540 LET UL=1
550 BORDER 8
560 WINDOW #2,1,11,1,25:PAPER #2,4:CLS #2
570 WINDOW #7,2,10,14,17:PAPER #7,4:PEN #7,8
580 WINDOW #1,2,10,3,5:PAPER #1,3:PEN #1,0
590 CLS #1:PRINT #1,"          HANGMAN          ";
600 WINDOW #3,12,20,1,25
610 WINDOW #4,12,20,1,15:PAPER #4,5:CLS #4
620 WINDOW #5,12,20,16,25:PAPER #5,6:CLS #5
630 WINDOW #6,2,10,7,23:PAPER #6,4
640 PLOT 0,0,3:DRAW 0,399:DRAW 639,399:DRAW 639,0:
DRAW 0,0:PLOT 350,0:DRAW 350,399
650 PEN #2,3
660 LOCATE #2,2,20:PRINT #2,"USED"
670 LOCATE #2,2,21:PRINT #2,"LETTERS:"
680 RETURN
690 REM ** BASE **
700 PAPER #3,7
710 LOCATE #3,2,23:PRINT #3,"       ";
720 RETURN
730 REM ** SIDE **
740 FOR Z=22 TO 3 STEP -1
750 LOCATE #3,2,Z:PRINT #3," ";
760 NEXT Z
770 PEN #3,6:LOCATE #3,3,22:PRINT #3,CHR$(213)
780 RETURN
790 REM ** TOP **
800 LOCATE #3,2,3:PRINT #3,"    ";
```

```
810 PEN #3,5:LOCATE #3,3,4:PRINT #3,CHR$(214)
820 PAPER #3,5:PEN #3,3
830 LOCATE #3,5,4:PRINT #3,CHR$(220)
840 LOCATE #3,5,5:PRINT #3,CHR$(221)
850 LOCATE #3,5,6:PRINT #3,CHR$(222)
860 RETURN
870 REM ** FACE **
880 LOCATE #3,5,4:PRINT #3,CHR$(225)
890 PEN #3,11
900 LOCATE #3,5,5:PRINT #3,CHR$(223);CHR$(224)
910 LOCATE #3,5,6:PRINT #3,CHR$(143);CHR$(143)
920 LOCATE #3,5,7:PRINT #3,CHR$(226);CHR$(227)
930 IF WR=9 THEN PLOT 500,320,3:DRAWR -4,-2:DRAWR
8,0:DRAWR -4,-2:GOTO 950
940 PLOT 500,320,4:DRAWR -4,-2:DRAWR 8,0:DRAWR -4,
-2
950 PLOT 520,320:DRAWR -4,-2:DRAWR 8,0:DRAWR -4,-2
960 PLOT 510,312:DRAWR 0,-2:DRAWR 4,0:DRAWR 0,2
970 IF WR=9 THEN PLOT 500,298:DRAWR 2,2:DRAWR 2,2:
DRAWR 4,2:DRAWR 6,0:DRAWR 4,-2:DRAWR 2,-2:DRAWR 2,
-2:GOTO 990
980 PLOT 500,304:DRAWR 2,-2:DRAWR 2,-2:DRAWR 4,-2:
DRAWR 6,0:DRAWR 4,2:DRAWR 2,2:DRAWR 2,2
990 RETURN
1000 REM ** BODY **
1010 PEN #3,12
1020 LOCATE #3,5,8:PRINT #3,CHR$(223);CHR$(224)
1030 FOR Z=9 TO 11
1040 LOCATE #3,5,Z:PRINT #3,CHR$(143);CHR$(143)
1050 NEXT Z
1060 PLOT 504,286,3:DRAWR 12,0:DRAWR 4,-2:DRAWR -2
0,0
1070 PEN #3,14
1080 LOCATE #3,5,12:PRINT #3,CHR$(143);CHR$(143)
1090 FOR Z=275 TO 235 STEP -20
1100 PLOT 510,Z,13:DRAWR 0,-2:DRAWR 4,0:DRAWR 0,2
1110 NEXT Z
1120 RETURN
1130 REM ** RIGHT ARM **
1140 PEN #3,12
1150 LOCATE #3,4,9:PRINT #3,CHR$(230)
1160 LOCATE #3,4,10:PRINT #3,CHR$(231)
1170 LOCATE #3,4,11:PRINT #3,CHR$(138)
1180 PEN #3,11:LOCATE #3,4,12:PRINT #3,CHR$(209)
1190 PAPER #3,14:LOCATE #3,5,12:PRINT #3,CHR$(211)
1200 PLOT 492,250,2:DRAWR 0,-26
1210 RETURN
1220 REM ** LEFT ARM **
1230 PEN #3,12:PAPER #3,5
1240 LOCATE #3,7,9:PRINT #3,CHR$(234)
```

```
1250 LOCATE #3,7,10:PRINT #3,CHR$(235)
1260 LOCATE #3,7,11:PRINT #3,CHR$(133)
1270 PEN #3,11:LOCATE #3,7,12:PRINT #3,CHR$(211)
1280 PAPER #3,14:LOCATE #3,6,12:PRINT #3,CHR$(209)
1290 PLOT 530,250,2:DRAWR 0,-26
1300 RETURN
1310 REM ** RIGHT LEG **
1320 PEN #3,14:PAPER #3,5
1330 LOCATE #3,5,13:PRINT #3,CHR$(143)
1340 LOCATE #3,4,14:PRINT #3,CHR$(230);CHR$(232)
1350 LOCATE #3,4,15:PRINT #3,CHR$(231);CHR$(233)
1360 PAPER #3,6
1370 LOCATE #3,4,16:PRINT #3,CHR$(138);CHR$(133)
1380 PEN #3,2:LOCATE #3,4,17:PRINT #3,CHR$(209);CH
R$(211)
1390 RETURN
1400 REM ** LEFT LEG **
1410 PEN #3,14:PAPER #3,5
1420 LOCATE #3,6,13:PRINT #3,CHR$(143)
1430 LOCATE #3,6,14:PRINT #3,CHR$(236);CHR$(234)
1440 LOCATE #3,6,15:PRINT #3,CHR$(237);CHR$(235)
1450 PAPER #3,6
1460 LOCATE #3,6,16:PRINT #3,CHR$(138);CHR$(133)
1470 PEN #3,2:LOCATE #3,6,17:PRINT #3,CHR$(209);CH
R$(211)
1480 RETURN
1490 REM ** SOLVED **
1500 LOCATE #2,2,14:PRINT #2,"WELL DONE"
1510 FOR Z=150 TO 1 STEP -1
1520 SOUND 1,Z,1:SOUND 2,Z+20,1:SOUND 3,Z+40,1
1530 NEXT Z
1540 GOTO 1710
1550 REM ** HANGED **
1560 LOCATE #2,2,14:PRINT #2,"HARD LUCK"
1570 FOR Z=1 TO 120
1580 LET R=1+INT(16*RND(1))
1590 INK 11,R
1600 LET R=1+INT(16*RND(1))
1610 INK 12,R
1620 LET R=1+INT(16*RND(1))
1630 INK 13,R
1640 LET R=1+INT(16*RND(1))
1650 INK 14,R
1660 SOUND 1,Z,1:SOUND 2,Z+20,1:SOUND 3,Z+40,1
1670 NEXT Z
1680 INK 11,26:INK 12,20:INK 13,8:INK 14,1
1690 PAPER #3,5:GOSUB 890
1700 PEN #2,3:LOCATE #2,ST,11:PRINT #2,W$:FOR Z=1
TO 1000:NEXT Z
1710 REM ** ANOTHER GO ? **
```

```
1720 LOCATE #7,1,3:PRINT #7," ANOTHER GO (Y/N)?";
1730 LET AG$=INKEY$
1740 IF AG$="Y" THEN GOSUB 1990:GOTO 50
1750 IF AG$="N" THEN CLS:MODE 1:END
1760 GOTO 1730
1770 RETURN
1780 REM ** USER DEFINED GRAPHICS **
1790 SYMBOL AFTER 220
1800 SYMBOL 220,8,8,8,8,8,8,8,8
1810 SYMBOL 221,8,28,20,20,20,34,34,34
1820 SYMBOL 222,34,34,34,34,20,20,28,8
1830 SYMBOL 223,&3,&7,&F,&1F,&3F,&3F,&7F,&7F
1840 SYMBOL 224,&C0,&E0,&F0,&F8,&FC,&FC,&FE,&FE
1850 SYMBOL 225,1,1,1,1,1,1,1,1
1860 SYMBOL 226,&7F,&7F,&3F,&3F,&1F,&F,&7,&3
1870 SYMBOL 227,&FE,&FE,&FC,&FC,&F8,&F0,&E0,&C0
1880 SYMBOL 228,&3,&7,&F,&F,&F,&F,&F,&F
1890 SYMBOL 229,&C0,&E0,&F0,&F0,&F0,&F0,&F0,&F0
1900 SYMBOL 230,1,1,1,1,3,3,3,3
1910 SYMBOL 231,7,7,7,7,15,15,15,15
1920 SYMBOL 232,&FF,&FF,&FF,&FF,&FE,&FE,&FE,&FE
1930 SYMBOL 233,&FC,&FC,&FC,&FC,&F8,&F8,&F8,&F8
1940 SYMBOL 234,&80,&80,&80,&80,&C0,&C0,&C0,&C0
1950 SYMBOL 235,&E0,&E0,&E0,&E0,&F0,&F0,&F0,&F0
1960 SYMBOL 236,&FF,&FF,&FF,&FF,&7F,&7F,&7F,&7F
1970 SYMBOL 237,&3F,&3F,&3F,&3F,&1F,&1F,&1F,&1F
1980 RETURN
1990 REM ** RE-INITIALIZE **
2000 PEN #2,3
2010 FOR Z=1 TO 26
2020 LET UL$(Z)=""
2030 NEXT Z
2040 LET UL=0:LET WR=0:LET COR=0
2050 CLS #4:CLS #5:CLS #6
2060 PLOT 0,0,3:DRAW 0,399:DRAW 639,399:DRAW 639,0
:DRAW 0,0:PLOT 350,0:DRAW 350,399
2070 INK 11,6
2080 LOCATE #2,2,20:PRINT #2,"USED"
2090 LOCATE #2,2,21:PRINT #2,"LETTERS:"
2100 RETURN
```

11. The Geographer

The Geographer is a graphical quiz requiring you to identify the locations of towns and cities on various maps which are displayed on the screen. The user is able to choose between two maps; one displaying the world and the other Great Britain. When a map has been selected, the main program will load it from the data cassette together with a database containing the location of the towns and cities to be used throughout the quiz.

Because the Geographer requires two maps as well as its own database, this chapter contains five programs:

1. Britain
2. The World
3. Create
4. Editor
5. The Geographer

These should be entered and saved in the correct sequence as indicated below.

Instructions

Unlike many of the other programs in this book, the Geographer does not take its information from EDBASE, but uses its own database. Accordingly, this chapter contains programs to create and edit the file.

To obtain a fully functional copy you will require two separate tapes which we will call Tape A and Tape B. You should enter and save the programs as follows:

Step 1 Enter the Geographer program and save it on to Tape A. Do not run the program at this stage as it will not function correctly.

Step 2 Enter the Editor program and save this on to Side 2 of Tape A. This is very important as it will be used to extend the somewhat limited database.

Step 3 Enter the first of the two maps, Britain, taking great care over the data values. When this program has been checked it should be run with Tape A, Side 1 in the Datacorder and the play and record keys depressed. The map will be constructed on the screen and then saved on to tape. When complete, repeat the process on Side 2 of the same tape.

Step 4 Repeat Step 3, using the second of the two maps, the World, positioning them after the two Britain maps.

Step 5 Finally, enter the Create program and with Tape B in the Datacorder type RUN so that the data can be transferred on to both sides of the tape.

When these five steps have been completed the tapes will contain:

Tape A (Side 1)	**Tape A (Side 2)**	**Tape B (Sides 1&2)**
Geographer	Editor	Database
Britain	Britain	
The World	The World	

These should be used as follows:

Using the Geographer

Place Tape A (Side 1) in the Datacorder and type LOAD " ". When loading is complete the program should be executed, and a message will appear on the screen enquiring which of the two maps is required. Enter B for Britain or W for the World, and the appropriate map will be loaded from tape and appear on the screen. Then Tape B, containing the database, should be placed in the Datacorder so that the information can be transferred into memory.

When all the loading has been completed the quiz will commence. A place name will appear at the bottom of the screen, and the user must then move the cursor (flashing dot) to the correct position on the map by using the cursor keys and press

enter when it is located at the correct place. After this process has been completed a total of ten times the game ends and a final score is displayed on the screen.

Using the Editor

The Editor Tape A (Side 2) is loaded in the same way as the Geographer, with the user selecting the required map and placing the database tape in the recorder when requested.

This program is a graphical database management program and is used to extend the limited database formed by the Create program. The mode of execution is similar to the Geographer program, with the user inputting the town or place name via the keyboard and then using the cursor keys to position the pointer at the appropriate location. Then the 'ENTER' key is pressed, placing the data into memory.

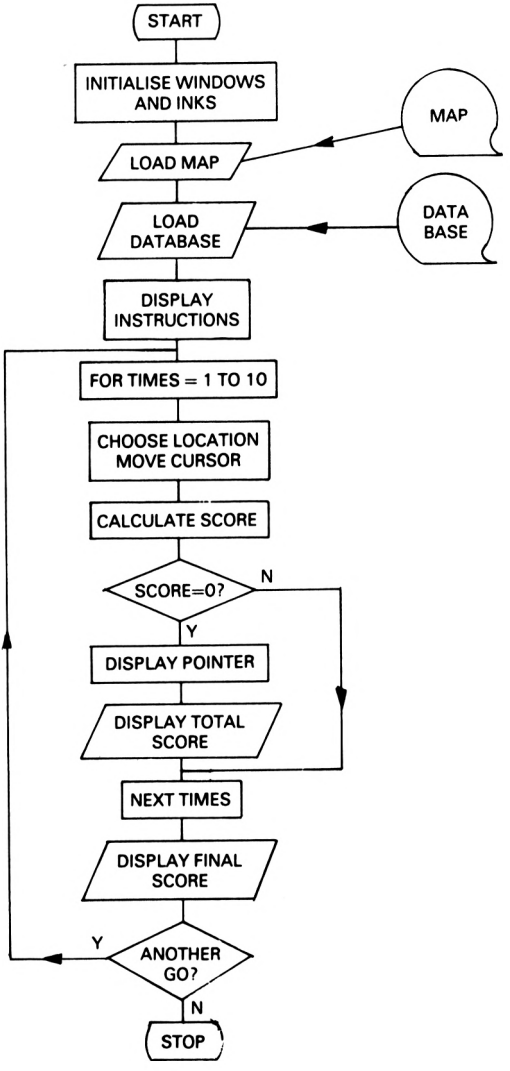
When editing has been completed, typing END as a response to PLACE NAME? will cause the database to be transferred to tape.

Geographer

This is the main program used to control the actual quiz.

Flowchart

The flowchart opposite represents the general operation of the program.


START
INITIALISE WINDOWS AND INKS
MAP
LOAD MAP
LOAD DATABASE
DATA BASE
DISPLAY INSTRUCTIONS
FOR TIMES = 1 TO 10
CHOOSE LOCATION MOVE CURSOR
CALCULATE SCORE
SCORE=0?
N
Y
DISPLAY POINTER
DISPLAY TOTAL SCORE
NEXT TIMES
DISPLAY FINAL SCORE
Y
ANOTHER GO?
N
STOP

Major variables

The following table indicates some of the major variables used in the program.

TYPE$	Flag indicating which map is being used
N$()	Name of location
PX()	X-co-ordinate of place on map
PY()	Y-co-ordinate of place on map
TY$()	Map on which place is stored
SCORE	Score
TIMES	Loop indicating the number of tests to be performed
POINTS	Number of points awarded for each test
SCORE$	String representing the score, for display purposes
X	X-co-ordinate of pixel cursor on screen
Y	Y-co-ordinate of pixel cursor on screen
MOV,DON DON1,COL	Flags concerned with the movement of the pixel cursor

Comments on the program

The program is structured, and the following notes show how the individual routines operate.

Lines 10-131
Define the windows and inks to be used throughout the program.

Lines 135-300
The user is asked which map he wishes to use, and this is then transferred from tape, followed by the database.

Lines 310-423
In this section the score and instructions are displayed on the right-hand side of the screen.

Lines 425-620
This is the main routine of the program, used to produce the quiz and compute the score.

Lines 670-740
This is the 'game over' routine to display the final score and ask the user if he wants another ago.

Lines 7000-7130
This routine is used to produce the arrow on the screen to indicate the position of a place that the user has failed to locate. This is quite complex, as a copy of the character square where the pointer is to go must be made so that it can be re-displayed.

Lines 8000-8095
A routine to move the pixel cursor around the screen.

The listing

The listing should be entered very carefully and then saved on to tape as outlined at the beginning 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.

```
10 MODE 1
15 INK 2,24
16 INK 3,1
20 INK 0,1
30 WINDOW #3,1,32,1,20
40 INK 1,18
50 PEN#4,2
70 PEN#3,1
80 PAPER#3,0
90 CLS
100 CLS#3
110 BORDER 1
120 WINDOW#4,1,40,21,25
130 WINDOW#5,33,40,1,20
131 PAPER#4,3:PAPER#5,3:CLS#4:CLS#5
135 LOCATE#3,14,9:PRINT#3,"THE GEOGRAPHER"
140 LOCATE#4,10,1:PRINT#4,"WHICH MAP - BRITAIN OR
THE WORLD ? (W/B)"
145 LET Q$=INKEY$:IF Q$="" THEN GOTO 145
146 IF Q$<>"W" AND Q$<>"B" THEN GOTO 145
147 LOCATE#4,15,3:PRINT#4," PRESS PLAY     "
150 IF Q$="W" THEN CLEAR:LOAD"!WORLD":LET TYPE$="W
"
160 IF Q$="B" THEN CLEAR:LOAD"!BRITAIN":LET TYPE$=
"B"
```

```
161 DIM N$(500),PX(500),PY(500),TY$(500)
162 DIM ST(16,16)
165 INK 2,0
170 BORDER 13
180 INK 3,13
190 CLS#4
200 CLS#5
210 PLOT 0,399,1:DRAWR 510,0:DRAWR 0,-318:DRAWR -5
10,0:DRAWR 0,318
215 LOCATE#4,2,1:PRINT#4,"INPUT DATA TAPE AND PRES
S PLAY"
220 OPENIN "!RECORDS"
230 LET I=1
240 IF EOF THEN GOTO 300
250 INPUT#9,N$(I)
260 INPUT#9,PX(I)
270 INPUT#9,PY(I)
271 INPUT#9,TY$(I)
275 IF EOF THEN GOTO 300
280 LET I=I+1
290 GOTO 240
300 CLOSEIN
310 LOCATE#5,1,1:PRINT#5,"MOVE"
320 PRINT#5,"CURSOR"
330 PRINT#5,"WITH"
340 PRINT#5,"JOYSTICK"
350 PRINT#5,"OR"
360 PRINT#5,"ARROW"
370 PRINT#5,"KEYS"
390 PRINT#5
400 PRINT#5,"ENTER"
410 PRINT#5,"TO"
420 PRINT#5,"PLACE."
421 PRINT#5:PRINT#5,"SCORE-"
422 LOCATE#5,1,14:PRINT#5,"00000"
423 LET SCORE=0
425 FOR TIMES=1 TO 10
430 CLS#4
435 FOR K=1 TO 20:LET Q$=INKEY$:NEXT K
440 LET J=INT(RND*I)+1:IF TY$(J)<>TYPE$ THEN GOTO
440
441 PRINT#4,"FIND ";N$(J);" ON THE MAP."
470 PRINT#4:PRINT#4,"MOVE CURSOR TO APPROPRIATE PO
SITION."
480 GOSUB 8000
490 LET POINTS=10
500 FOR D=1 TO 10
510 IF X-PX(J)>D OR X-PX(J)<-D OR Y-PY(J)>D OR Y-P
Y(J)<-D THEN LET POINTS=POINTS-1
520 NEXT D
```

```
530 CLS#4
540 PRINT#4:PRINT#4,"YOU HAVE SCORED ";POINTS;"POI
NTS."
541 IF POINTS=0 THEN GOSUB 7000
545 FOR S=1 TO 400:NEXT
550 LET SCORE=SCORE+POINTS
560 LET SCORE$=STR$(SCORE)
570 IF LEN(SCORE$)=2 THEN LET SCORE$="0000"+RIGHT$
(SCORE$,1)
580 IF LEN(SCORE$)=3 THEN LET SCORE$="000"+RIGHT$(
SCORE$,2)
590 IF LEN(SCORE$)=4 THEN LET SCORE$="00"+RIGHT$(S
CORE$,3)
600 IF LEN(SCORE$)=5 THEN LET SCORE$="0"+RIGHT$(SC
ORE$,4)
610 LOCATE#5,1,14:PRINT#5,SCORE$
620 NEXT TIMES
670 CLS#4
680 PRINT#4
690 PRINT#4,"         TEST OVER"
700 PRINT#4,"YOU SCORED ";SCORE$;" OVERALL."
710 PRINT#4,"  ANOTHER GO? (Y OR N)"
715 LET Z$=INKEY$
720 IF Z$="Y" THEN GOTO 422
730 IF Z$="N" THEN END
740 GOTO 715
7000 REM **POINTER**
7001 LET N=0:LET M=0
7010 FOR K=PY(J)-11 TO PY(J)+3
7015 LET M=M+1
7016 LET N=0
7020 FOR L=PX(J)+3 TO PX(J)+18
7025 LET N=N+1
7030 LET ST(N,M)=TEST(L,K)
7040 NEXT L,K
7050 PLOT 1000,1000,3:MOVE PX(J)+3,PY(J)+3:TAG
7055 PRINT CHR$(242);
7060 TAGOFF
7070 PRINT#4:PRINT#4,"THIS IS ";N$(J)
7080 FOR T=1 TO 1000:NEXT
7085 LET N=0:LET M=0
7090 FOR K=PY(J)-11 TO PY(J)+3
7095 LET M=M+1
7096 LET N=0
7100 FOR L=PX(J)+3 TO PX(J)+18
7105 LET N=N+1
7110 PLOT L,K,ST(N,M)
7120 NEXT L,K
7130 RETURN
8000 LET X=320
```

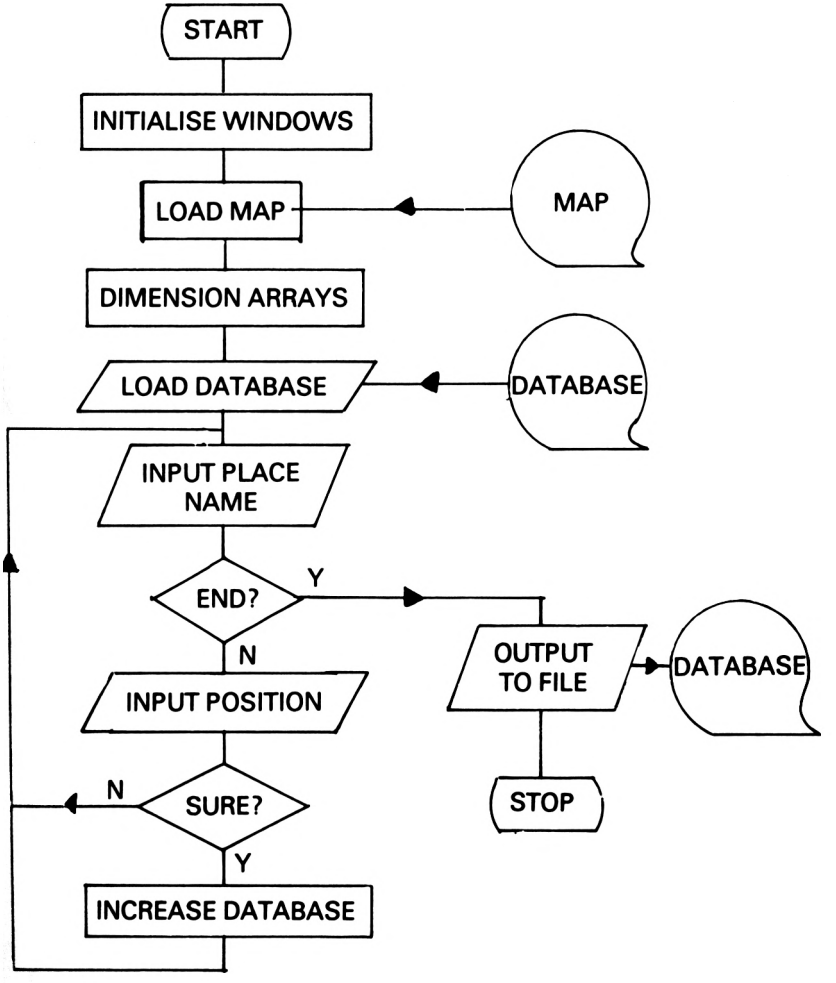
```
8001 LET MOV=0
8005 FOR K=1 TO 20:LET Q$=INKEY$:NEXT K
8010 LET Y=200
8015 LET COL=TEST(X,Y)
8020 PLOT X,Y,2
8030 IF INKEY(18)=0 THEN GOSUB 8090:RETURN
8040 IF (INKEY(0)=0 OR INKEY(72)=0) AND Y<397 THEN
 GOSUB 8090:LET DON=1:LET Y=Y+1
8050 IF (INKEY(2)=0 OR INKEY(73)=0) AND Y>83 THEN
GOSUB 8090:LET DON=1:LET Y=Y-1
8060 IF (INKEY(8)=0 OR INKEY(74)=0) AND X>0 AND DO
N=0 THEN GOSUB 8090:LET X=X-1
8065 IF (INKEY(8)=0 OR INKEY(74)=0) AND X>0 AND DO
N=1 THEN LET X=X-1
8070 IF (INKEY(1)=0 OR INKEY(75)=0) AND X<509 AND
DON=0 THEN GOSUB 8090:LET X=X+1
8075 IF (INKEY(1)=0 OR INKEY(75)=0) AND X<509 AND
DON=1 THEN LET X=X+1
8079 LET DON=0:LET DON1=0
8080 IF MOV=0 THEN GOTO 8020 ELSE LET MOV=0:GOTO 8
015
8085 RETURN
8090 LET MOV=1:PLOT X,Y,COL
8095 RETURN
```

Editor

This program enables the user to extend the database. This is done graphically, hence the program is very similar to the previous one.

Flowchart

The flowchart opposite represents the general operation of the program.


START
INITIALISE WINDOWS
LOAD MAP
MAP
DIMENSION ARRAYS
LOAD DATABASE
DATABASE
INPUT PLACE NAME
END?
Y
N
INPUT POSITION
SURE?
N
Y
INCREASE DATABASE
OUTPUT TO FILE
DATABASE
STOP

Major variables

The following table shows some of the major variables used throughout the program and should help you to understand how the program operates.

TYPE$	Flag indicating which map is being used
N$()	Name of location
PX()	X-co-ordinate of location on map
PY()	Y-co-ordinate of location on map
TY$()	Map on which location is stored
MON,DON	Flags used with the movement of the pixel cursor

Comments on the program

The following notes show the operation of the individual routines in the program.

Lines 10-136
These lines define the windows and inks to be used throughout the program.

Lines 140-160
The user is asked which map he wishes to use, and this is then transferred from tape.

Lines 161-210
The arrays to be used throughout the program are dimensioned and the screen display is completed.

Lines 220-300
This routine loads the database from Tape B.

Lines 310-430
The instructions are displayed in the window on the right-hand side of the screen.

Lines 435-525
This is the main routine used to extend the database. The user will be asked the name of a place to be added and then instructed to

move the black pixel cursor to the appropriate position on the screen. When in the correct location the ENTER key is pressed and the user is asked to verify the position after which the information will be added to the database.

Lines 540-660
When the user has finished extending the database he must type END, and then this routine will output the database on to the cassette tape.

Lines 8000-8095
A routine to move the black pixel cursor around the screen by using a joystick or the cursor keys.

The listing

The listing should be entered very carefully and then saved onto tape as outlined at the beginning of this chapter.

```
10 MODE 1
15 INK 2,24
16 INK 3,1
20 INK 0,1
30 WINDOW #3,1,32,1,20
40 INK 1,18
50 PEN#4,2
70 PEN#3,1
80 PAPER#3,0
90 CLS
100 CLS#3
110 BORDER 1
120 WINDOW#4,1,40,21,25
130 WINDOW#5,33,40,1,20
131 PAPER#4,3:PAPER#5,3:CLS#4:CLS#5
135 LOCATE#3,14,9:PRINT#3,"THE GEOGRAPHER"
136 LOCATE #3,16,11:PRINT#3,"PLACE MAKER"
140 LOCATE#4,2,1:PRINT#4,"WHICH MAP - BRITAIN OR T
HE WORLD ? (W/B)"
145 LET Q$=INKEY$:IF Q$="" THEN GOTO 145
146 IF Q$<>"W" AND Q$<>"B" THEN GOTO 145
147 LOCATE#4,2,1:PRINT#4,"PRESS PLAY ON DATA CASSE
TTE."
150 IF Q$="W" THEN CLEAR:LOAD"!WORLD":LET TYPE$="W
"
160 IF Q$="B" THEN CLEAR:LOAD"!BRITAIN":LET TYPE$=
"B"
```

```
161 DIM N$(500),PX(500),PY(500),TY$(500)
165 INK 2,0
170 BORDER 13
180 INK 3,13
190 CLS#4
200 CLS#5
210 PLOT 0,399,1:DRAWR 510,0:DRAWR 0,-318:DRAWR -5
10,0:DRAWR 0,318
215 LET I=0
220 OPENIN "!RECORDS"
230 LET I=0
240 IF EOF THEN GOTO 300
250 INPUT#9,N$(I)
260 INPUT#9,PX(I)
270 INPUT#9,PY(I)
271 INPUT#9,TY$(I)
275 IF EOF THEN GOTO 300
280 LET I=I+1
290 GOTO 240
300 CLOSEIN
310 LOCATE#5,1,1:PRINT#5,"MOVE"
320 PRINT#5,"CURSOR"
330 PRINT#5,"WITH"
340 PRINT#5,"JOYSTICK"
350 PRINT#5,"OR"
360 PRINT#5,"ARROW"
370 PRINT#5,"KEYS"
390 PRINT#5
400 PRINT#5,"ENTER"
410 PRINT#5,"TO"
420 PRINT#5,"PLACE."
430 CLS#4
435 FOR K=1 TO 20:LET Q$=INKEY$:NEXT K
440 INPUT#4,"PLACE NAME ";N$(I+1)
445 IF N$(I+1)="END" THEN GOTO 540
446 IF LEN(N$(I+1))>12 THEN CLS#$:GOTO 440
450 FOR K=1 TO I
460 IF N$(K)=N$(I+1) THEN GOTO 430
465 NEXT K
470 PRINT#4:PRINT#4,"MOVE CURSOR TO APPROPRIATE PO
SITION."
480 GOSUB 8000
490 CLS#4
500 LOCATE#4,12,3:PRINT#4,"ARE YOU SURE?"
510 IF INKEY(46)=0 THEN GOTO 430
520 IF INKEY(43)=0 THEN LET I=I+1:LET TY$(I)=TYPE$
:GOTO 430
525 GOTO 510
540 CLS#4
550 LOCATE#4,2,2:PRINT#4,"PLACE DATA TAPE IN DECK
```

```
AT CORRECT       POSITION AFTER MAPS."
560 PRINT#4:PRINT#4,"HIT ENTER TO BEGIN RECORDING.
"
570 IF INKEY(18)<>0 THEN GOTO 570
590 OPENOUT "!RECORDS"
600 FOR K=1 TO I
610 PRINT#9,N$(K)
620 PRINT#9,PX(K)
630 PRINT#9,PY(K)
640 NEXT K
650 CLOSEOUT
660 GOTO 430
8000 LET PX(I+1)=320
8001 LET MOV=0
8005 FOR K=1 TO 20:LET Q$=INKEY$:NEXT K
8010 LET PY(I+1)=200
8015 LET COL=TEST(PX(I+1),PY(I+1))
8020 PLOT PX(I+1),PY(I+1),2
8030 IF INKEY(18)=0 THEN GOSUB 8090:RETURN
8040 IF (INKEY(0)=0 OR INKEY(72)=0) AND PY(I+1)<39
7 THEN GOSUB 8090:LET DON=1:LET PY(I+1)=PY(I+1)+1
8050 IF (INKEY(2)=0 OR INKEY(73)=0) AND PY(I+1)>83
 THEN GOSUB 8090:LET DON=1:LET PY(I+1)=PY(I+1)-1
8060 IF (INKEY(8)=0 OR INKEY(74)=0) AND PX(I+1)>0
AND DON=0 THEN GOSUB 8090:LET PX(I+1)=PX(I+1)-1
8065 IF (INKEY(8)=0 OR INKEY(74)=0) AND PX(I+1)>0
AND DON=1 THEN LET PX(I+1)=PX(I+1)-1
8070 IF (INKEY(1)=0 OR INKEY(75)=0) AND PX(I+1)<50
9 AND DON=0 THEN GOSUB 8090:LET PX(I+1)=PX(I+1)+1
8075 IF (INKEY(1)=0 OR INKEY(75)=0) AND PX(I+1)<50
9 AND DON=1 THEN LET PX(I+1)=PX(I+1)+1
8079 LET DON=0:LET DON1=0
8080 IF MOV=0 THEN GOTO 8020 ELSE LET MOV=0:GOTO 8
015
8085 RETURN
8090 LET MOV=1:PLOT PX(I+1),PY(I+1),COL
8095 RETURN
```

Britain

This program creates a map of Great Britain on the screen and then, by using the command:

```
SAVE "BRITAIN",B,&C000
```

stores the contents of the memory concerned with the screen directly on to tape. Former Spectrum owners may have used the

similar command:

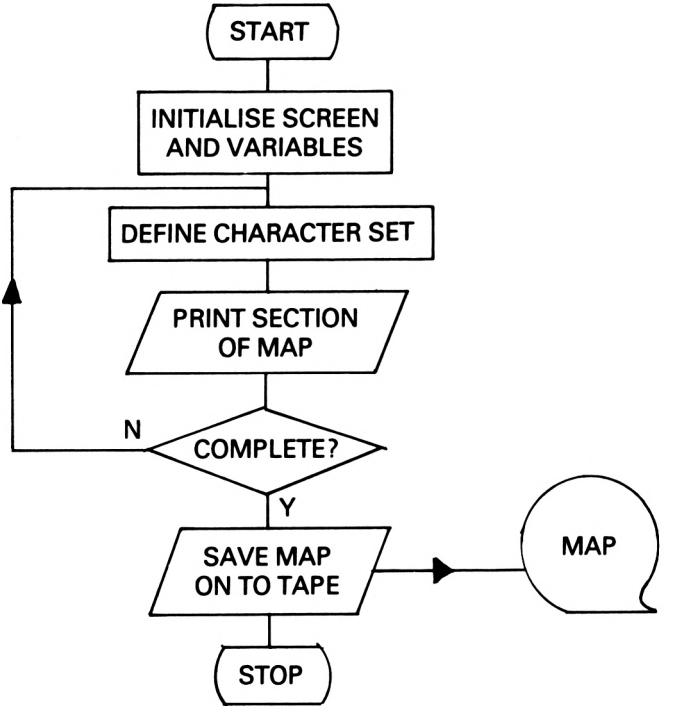
```
SAVE "BRITAIN"SCREEN$
```

which also performs this task.

The program achieves its objectives by repeatedly redefining the character set as the outline of the country and then displaying the characters on the screen. This gives the coastline a definite shape, with the inland areas being made up of complete blocks (CHR$(143)).

Flowchart

The flowchart represents the general operation of the program.



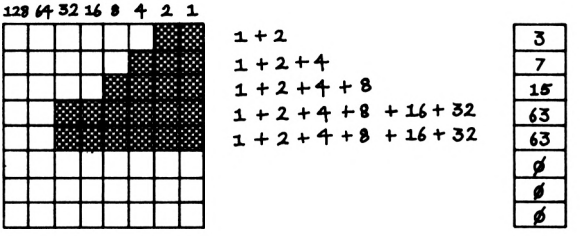
Comments on the program

The program can be divided into two routines:

1. Lines 10-1480 and 9000-9060
2. Lines 1490-1520

Lines 10-1480
This section of program contains the data lines for use in the routine at line 9000, together with PRINT statements to display the map on the screen.

Lines 9000-9060
In this routine, the characters A to Z are redefined to appear as sections of the coastline. Each character is defined by reading the eight bytes of data that make up the character. The configuration is as follows.



Therefore, to define this character we would use the statements:

```
SYMBOL AFTER X
SYMBOL X,3,7,15,63,63,0,0,0
```

where 'X' is any integer in the range 32-255. The 'symbol after' command tells the computer where in the character set the re-defining is to commence.

The listing

The listing should be entered very carefully and then saved on to tape as instructed as the beginning of this chapter.

```
1 MODE 1:CLEAR:LET F=90
2 PAPER 0:BORDER 1
3 PEN 1
4 INK 0,1
5 INK 1,18
6 CLS
7 RESTORE
8 DIM A(8)
9 SYMBOL AFTER 65
10 DATA 0,0,0,1,3,3,7,7
20 DATA 1,3,63,255,255,255,255,255
30 DATA 128,192,192,192,128,0,0,0
40 DATA 0,6,15,30,60,125,127,255
50 DATA 0,0,0,0,124,254,255,255
60 DATA 0,0,0,0,0,0,128,224
70 DATA 7,39,115,249,248,112,96,0
80 DATA 254,252,240,224,128,0,0,0
85 DATA 1,7,15,7,49,121,125,255
90 DATA 255,255,255,255,255,255,255,255
100 DATA 255,255,255,255,255,255,252,248
110 DATA 254,255,254,248,240,0,0,0
120 DATA 0,0,0,0,0,1,0,0
130 DATA 0,192,97,115,1,225,195,1
140 DATA 120,56,252,248,240,240,248,248
150 DATA 255,127,63,63,127,127,255,255
160 DATA 255,255,255,255,253,252,252,253
170 DATA 240,192,192,224,240,224,0,63
180 DATA 0,0,0,0,0,0,0,240
190 DATA 0,0,24,48,0,0,0,0
200 DATA 0,0,24,60,60,24,8,0
210 DATA 249,109,7,3,1,7,31,63
220 DATA 255,255,255,255,255,255,255,255
230 DATA 249,251,251,255,255,255,255,255
240 DATA 127,255,255,255,255,255,255,255
250 DATA 252,254,255,255,255,255,255,255
260 GOSUB 9000
270 LOCATE 13,1:PRINT"ABCDEF":LOCATE 13,2:PRINT"GH
IJKL":LOCATE 12,3:PRINT"MNOPQRS":LOCATE 12,4:PRINT
"TUVWXYZ"
280 DATA 0,0,128,192,224,224,224,224
290 DATA 64,192,128,0,0,6,14,12
300 DATA 31,63,127,127,63,127,255,255
310 DATA 192,192,128,224,224,192,192,128
320 DATA 12,28,0,0,0,0,0,0
330 DATA 31,31,15,15,31,31,123,123
340 DATA 255,255,254,254,248,240,224,192
350 DATA 192,128,0,0,0,0,0,0
360 DATA 0,0,0,0,16,49,120,255
370 DATA 0,0,0,66,239,255,254,254
380 DATA 0,0,0,0,3,7,13,3
```

```
390 DATA 59,51,119,231,230,224,198,207
400 DATA 255,255,127,63,63,63,159,223
410 DATA 255,255,255,255,195,224,252,253
420 DATA 192,192,224,224,192,0,0,0
430 DATA 0,0,0,0,0,3,15,31
440 DATA 0,0,0,0,0,0,128,128
450 DATA 1,3,7,15,63,63,31,15
471 DATA 252,249,255,255,255,255,255,255
472 DATA 0,0,126,255,255,255,255,255
473 DATA 7,7,15,143,134,240,224,224
474 DATA 159,159,31,63,63,63,63,127
480 DATA 192,224,240,240,240,224,192,224
490 DATA 63,127,63,127,127,63,31,79
500 DATA 192,225,255,255,255,255,255,255
510 DATA 7,255,255,255,255,255,255,255
520 GOSUB 9000
530 LOCATE 19,4:PRINT"A":LOCATE 13,5:PRINT"BC";CHR
$(143);CHR$(143);CHR$(143);CHR$(143);"D":LOCATE 13
,6:PRINT"EF";CHR$(143);CHR$(143);CHR$(143);"GH"
540 LOCATE 10,7:PRINT"IJ KLM";CHR$(143);"NO":LOCAT
E 7,8:PRINT"PQR";CHR$(143);"STUV";CHR$(143);CHR$(1
43);CHR$(143);CHR$(143);"W":LOCATE 7,9:PRINT"XYZ";
CHR$(143);CHR$(143);CHR$(143)
550 DATA 248,240,240,248,248,128,204,248
560 DATA 63,127,255,255,255,255,111,103
570 DATA 240,252,248,248,252,252,254,254
580 DATA 0,0,0,1,7,15,15,15
590 DATA 103,127,255,255,255,255,255,255
600 DATA 240,224,224,224,240,224,192,192
610 DATA 221,99,66,0,0,0,0,0
620 DATA 188,56,112,0,0,0,0,0
630 DATA 127,31,31,63,255,255,255,255
640 DATA 252,252,252,252,254,254,254,254
650 DATA 15,31,15,7,0,0,0,0
660 DATA 255,255,255,255,254,254,3,254
670 DATA 255,240,224,192,128,128,128,128
680 DATA 30,31,62,252,248,240,96,0
690 DATA 31,31,15,15,7,7,7,3
700 DATA 255,255,255,255,255,255,255,255
710 DATA 254,254,254,254,254,254,255,254
720 DATA 0,1,3,7,31,63,48,7
730 DATA 255,255,255,255,255,255,63,255
740 DATA 254,254,255,255,255,255,254,252
750 DATA 15,31,31,63,127,255,255,255
760 DATA 248,252,254,254,255,255,255,255
770 DATA 0,0,15,111,255,127,63,255
780 DATA 31,63,63,255,255,255,255,255
790 DATA 252,252,254,255,255,254,252,248
800 DATA 0,0,3,1,1,0,0,7
810 GOSUB 9000
```

```
820 LOCATE 13,9:PRINT"AB";CHR$(143);CHR$(143);CHR$
(143);CHR$(143);"C":LOCATE 6,10:PRINT"DE";:FOR I=1
 TO 5:PRINT CHR$(143);:NEXT:PRINT"FGHI";CHR$(143);
CHR$(143)"J"
830 LOCATE 6,11:PRINT"KL";CHR$(143);CHR$(143);CHR$
(143);CHR$(143);"M  NOP";CHR$(143);"Q":LOCATE 6,12
:PRINT"RS";CHR$(143);CHR$(143);CHR$(143);"T    U";
CHR$(143);CHR$(143);CHR$(143);"V"
840 LOCATE 5,13:PRINT"WX";CHR$(143);CHR$(143);CHR$
(143);CHR$(143);"Y Z"
850 DATA 0,0,192,207,223,159,63,255
860 DATA 0,0,1,129,193,248,250,255
870 DATA 255,255,255,255,255,127,127,255
880 DATA 255,255,255,255,255,221,225,252
890 DATA 0,0,0,192,224,240,240,224
900 DATA 127,255,243,7,31,127,252,112
910 DATA 255,255,255,255,255,63,127,255
920 DATA 255,255,255,255,255,255,255,232
921 DATA 255,255,255,255,254,254,254,252
922 DATA 224,192,128,0,0,0,0,0
930 DATA 31,63,63,1,1,3,7,7
940 DATA 254,254,255,254,254,254,255,255
950 DATA 7,15,6,0,0,0,0,0
960 DATA 255,255,3,0,0,0,0,0
970 DATA 247,225,192,0,0,0,0,0
980 DATA 255,255,112,0,0,0,0,0
990 DATA 228,192,0,0,0,0,0,0
1000 DATA 252,120,112,16,0,0,0,0
1010 DATA 0,0,0,0,0,12,15,7
1020 DATA 0,0,0,0,0,15,255,255
1030 DATA 3,1,3,51,255,255,255,255
1040 DATA 255,254,254,252,248,249,255,255
1050 DATA 0,0,0,32,127,255,255,255
1060 DATA 0,0,0,0,192,240,248,248
1070 DATA 1,3,7,15,4,0,0,0
1080 DATA 255,255,191,63,127,255,255,3
1090 GOSUB 9000
1100 LOCATE 14,13:PRINT"ABC";CHR$(143);CHR$(143);C
HR$(143);"DE":LOCATE 5,14:PRINT"FG";CHR$(143);CHR$
(143);"HIJ K";:FOR I=1 TO 6:PRINT CHR$(143);:NEXT:
PRINT"L"
1110 LOCATE 5,15:PRINT"MNOPQRSTU";:FOR I=1 TO 6:PR
INT CHR$(143);:NEXT:PRINT"VWX":LOCATE 11,16:PRINT"
YZ"
1120 DATA 255,255,255,255,255,255,63,127
1130 DATA 240,248,252,252,252,252,248,248
1140 DATA 0,1,3,7,3,0,0,0
1150 DATA 127,255,253,249,3,7,0,0
1160 DATA 255,255,255,255,255,255,1,7
1170 DATA 248,248,252,252,248,192,192,192
```

```
1180 DATA 0,0,0,0,0,0,3,31
1190 DATA 0,3,3,19,51,255,255,255
1200 DATA 0,0,128,254,255,255,255,255
1210 DATA 3,3,15,31,255,255,255,255
1220 DATA 255,255,255,255,255,254,128,252
1230 DATA 255,255,255,222,206,128,0,0
1240 DATA 0,0,3,7,127,111,15,31
1250 DATA 63,255,255,255,255,255,158,152
1260 DATA 255,255,255,255,255,191,31,31
1270 DATA 255,255,255,255,255,255,254,252
1280 DATA 255,255,255,255,255,63,15,15
1290 DATA 255,255,255,255,255,255,255,255
1300 DATA 255,255,255,255,255,255,255,135
1310 DATA 240,248,252,254,255,255,252,248
1320 DATA 31,14,0,0,0,0,0,0
1330 DATA 16,0,0,0,0,0,0,0
1340 DATA 15,7,7,3,1,0,0,0
1350 DATA 252,248,252,252,224,0,0,0
1360 DATA 255,62,0,0,0,0,0,0
1370 DATA 48,126,127,255,252,32,0,0
1380 GOSUB 9000
1390 LOCATE 13,16:PRINT"A";:FOR I=1 TO 8:PRINT CHR
$(143);:NEXT:PRINT"B":LOCATE 12,17:PRINT"CDE";:FOR
 I=1 TO 7:PRINT CHR$(143);:NEXT:PRINT"F"
1400 LOCATE 11,18:PRINT"GHIJ";:FOR I=1 TO 5:PRINT
CHR$(143);:NEXT:PRINT"KL":LOCATE 10,19:PRINT"MNOPQ
RS";:FOR I=1 TO 4:PRINT CHR$(143);:NEXT:PRINT"T"
1410 LOCATE 10,20:PRINT"UVWX YZ";
1420 DATA 63,63,31,0,0,0,0,0
1430 DATA 255,255,255,127,0,0,0,0
1440 DATA 255,255,255,254,220,0,0,0
1450 DATA 248,240,192,0,0,0,0,0
1460 LET F=68
1470 GOSUB 9000
1480 PRINT"ABCD"
1490 SPEED WRITE 1
1500 CLEAR
1510 SAVE "!BRITAIN",B,&C000,16383,&C000
1520 END
9000 FOR I=65 TO F
9010 FOR J=1 TO 8
9020 READ A(J)
9030 NEXT J
9040 SYMBOL I,A(1),A(2),A(3),A(4),A(5),A(6),A(7),A
(8)
9050 NEXT I
9060 RETURN
```

The World

This program is essentially the same as the previous one, except that it will produce a map of the world.

The listing

Enter the program with great care and save on to tape as outlined at the beginning of this chapter.

```
10 CLS
20 INK 0,1
30 INK 1,18
40 LET F=90
100 DATA 63,127,255,255,255,127,63,31
110 DATA 64,192,192,192,193,135,143,7
120 DATA 63,63,63,127,255,255,255,255
130 DATA 240,240,248,240,96,160,64,0
140 DATA 3,1,3,3,7,15,15,7
150 DATA 252,252,248,240,240,248,248,252
160 DATA 0,0,24,62,31,15,7,7
170 DATA 0,0,0,0,128,224,192,192
180 DATA 31,63,127,127,127,60,48,0
190 DATA 7,7,15,63,63,31,31,15
200 DATA 255,255,254,254,252,252,254,255
210 DATA 7,7,15,63,63,31,15,7
220 DATA 252,252,252,248,252,254,252,252
230 DATA 0,0,0,0,0,0,0,60
240 DATA 224,112,0,0,0,0,0,0
250 DATA 15,15,7,31,31,63,29,8
260 DATA 255,254,255,255,254,254,231,2
270 DATA 7,7,1,0,0,0,0,0
280 DATA 255,255,255,63,31,7,1,1
290 DATA 252,252,254,254,252,252,248,248
300 DATA 1,3,3,15,7,63,63,255
310 DATA 252,255,254,254,252,254,255,255
320 DATA 0,0,0,0,0,128,192,192
330 DATA 61,127,127,31,31,15,15,31
340 DATA 128,128,192,230,207,207,143,7
350 DATA 0,0,28,62,254,252,248,248
360 DATA 0,0,0,15,31,127,255,255
370 DATA 0,0,0,192,224,250,254,255
380 DATA 255,127,127,127,15,15,3,1
390 DATA 255,255,255,255,254,252,255,254
```

```
400 DATA 224,224,128,0,0,0,0,0
410 DATA 0,0,0,0,3,103,225,225
420 DATA 1,3,51,55,255,255,255,255
430 DATA 192,248,255,255,255,255,255,255
440 DATA 0,0,192,208,252,253,255,255
450 DATA 0,0,0,0,6,143,255,255
460 DATA 0,0,0,0,0,198,254,255
470 DATA 0,0,0,0,0,0,8,191
480 DATA 3,15,63,127,255,127,127,127
490 DATA 230,255,255,255,255,255,255,255
500 DATA 0,128,252,255,255,255,255,255
510 DATA 0,0,62,255,255,255,255,255
520 DATA 0,0,112,252,254,255,255,255
530 DATA 15,7,7,1,0,224,241,251
540 DATA 248,240,243,199,7,31,255,255
550 DATA 0,96,252,252,255,255,255,255
560 DATA 255,255,31,31,15,135,192,192
570 DATA 252,254,255,255,255,255,127,63
580 DATA 0,0,0,128,192,192,192,128
590 DATA 31,31,31,15,15,31,63,31
600 DATA 254,255,255,255,255,255,255,254
610 DATA 0,0,128,128,128,192,128,0
620 DATA 0,0,0,3,15,15,31,255
630 DATA 0,0,128,224,248,255,255,255
640 DATA 0,0,0,0,0,0,192,192
650 DATA 3,7,7,15,63,63,127,255
660 DATA 127,127,255,127,63,63,31,31
670 DATA 255,255,255,255,255,255,252,248
680 DATA 224,224,240,224,192,0,0,0
690 DATA 63,255,255,255,127,63,63,15
700 DATA 0,128,192,192,192,192,128,0
710 DATA 31,63,127,127,127,63,31,31
720 DATA 255,255,255,255,255,242,224,224
730 DATA 254,248,224,192,0,0,0,0
740 DATA 0,0,0,0,16,54,127,126
750 DATA 3,3,7,15,31,31,31,31
760 DATA 255,255,255,255,255,255,243,247
765 DATA 192,128,192,227,199,135,31,63
770 DATA 1,1,195,231,255,255,255,255
780 DATA 1,1,195,231,255,255,255,255
790 DATA 63,63,63,47,15,7,3,1
800 DATA 255,255,255,255,254,252,192,128
810 DATA 255,255,255,255,63,31,7,3
820 DATA 252,252,240,248,252,252,248,224
830 DATA 0,0,0,1,7,15,7,7
840 DATA 0,0,0,0,192,224,240,252
850 DATA 31,31,15,3,0,0,0,0
860 DATA 240,240,240,224,192,64,0,0
870 DATA 63,63,127,127,255,255,255,239
880 DATA 247,247,231,239,207,207,199,199
```

```
890 DATA 255,255,255,255,255,255,252,240
900 DATA 255,254,248,248,224,128,0,0
910 DATA 15,7,7,3,0,0,0,0
920 DATA 255,255,255,255,63,31,31,15
930 DATA 224,248,254,255,255,255,255,255
940 DATA 7,15,15,15,159,159,255,255
950 DATA 192,224,248,252,252,248,248,240
960 DATA 1,3,3,1,49,241,243,243
980 DATA 192,224,224,224,192,224,224,240
990 DATA 239,239,102,6,4,0,0,0
1000 DATA 143,15,7,15,15,63,127,255
1010 DATA 224,224,240,224,192,192,128,192
1020 DATA 31,31,15,7,7,15,7,3
1030 DATA 248,240,224,224,240,240,224,192
1040 DATA 243,240,192,1,0,0,0,0
1050 DATA 252,249,227,131,7,255,127,63
1060 DATA 31,255,255,255,255,255,255,255
1070 DATA 252,248,128,192,192,248,252,248
1080 DATA 255,127,127,63,127,31,15,15
1090 DATA 255,255,252,228,192,192,128,0
1100 DATA 224,128,128,0,0,0,0,0
1110 DATA 0,0,28,63,127,127,127,63
1120 DATA 63,63,127,255,255,255,252,248
1130 DATA 255,255,255,255,253,124,30,15
1140 DATA 255,255,255,255,127,30,24,0
1150 DATA 255,255,255,255,230,192,14,31
1160 DATA 255,255,255,255,127,63,255,255
1170 DATA 254,252,240,228,224,192,128,128
1180 DATA 127,127,63,63,31,31,15,3
1190 DATA 255,254,252,252,252,248,240,240
1200 DATA 31,7,3,0,0,0,3,7
1210 DATA 224,192,128,0,96,240,253,255
1220 DATA 3,3,1,0,0,224,248,255
1230 DATA 128,0,0,0,0,34,242,247
1240 DATA 31,31,15,15,7,3,199,255
1250 DATA 255,231,242,240,248,240,224,224
1260 DATA 255,127,127,63,31,31,15,7
1270 DATA 255,255,255,252,252,248,252,255
1280 DATA 249,144,16,0,0,0,0,206
1290 DATA 240,224,128,0,0,0,0,0
1300 DATA 15,15,31,31,31,31,63,63
1310 DATA 255,255,231,225,240,254,255,255
1320 DATA 255,255,255,255,127,12,192,192
1330 DATA 255,255,255,255,159,31,31,15
1340 DATA 255,255,255,255,254,248,240,224
1350 DATA 3,1,0,0,0,0,0,0
1360 DATA 255,255,127,31,23,3,0,0
1370 DATA 255,255,255,255,255,255,135,0
1380 DATA 0,144,248,248,252,252,254,63
1390 DATA 0,0,0,0,0,0,92,255
```

```
1400 DATA 0,0,0,0,0,0,192,240
1410 DATA 63,63,63,127,127,127,127,63
1420 DATA 255,207,227,241,249,248,252,254
1430 DATA 255,255,254,192,0,0,0,0
1440 DATA 128,128,0,0,0,0,0,0
1450 DATA 15,15,7,7,15,15,7,3
1460 DATA 255,254,254,254,248,248,240,192
1470 DATA 255,63,31,7,7,3,0,0
1480 DATA 255,252,252,254,255,254,124,124
1490 DATA 224,224,64,0,0,0,0,0
1500 DATA 63,31,127,63,31,63,63,31
1510 DATA 248,248,248,252,255,255,255,255
1520 DATA 0,0,0,0,0,128,192,224
1530 DATA 63,63,31,7,0,0,0,0
1540 DATA 255,255,255,255,255,63,31,3
1550 DATA 192,224,224,224,224,192,192,128
1560 DATA 3,1,0,0,0,0,0,0
1570 DATA 192,128,0,0,0,0,0,0
1580 DATA 63,31,31,124,248,248,124,126
1590 DATA 63,31,15,15,15,15,31,15
1600 DATA 252,254,255,255,255,255,255,255
1610 DATA 0,0,128,240,248,248,252,252
1620 DATA 255,127,63,63,31,31,15,3
1630 DATA 254,252,252,240,240,248,240,240
1640 DATA 62,62,31,15,6,0,0,0
1650 DATA 15,7,7,1,0,0,0,0
1670 DATA 248,248,240,240,192,192,224,192
1680 DATA 15,7,7,3,3,1,1,0
1690 DATA 248,248,248,248,240,240,225,193
1700 DATA 0,0,8,56,120,248,240,240
1710 DATA 0,0,0,0,0,0,3,7
1720 DATA 0,0,1,3,3,127,255,255
1730 DATA 0,48,248,252,252,255,255,255
1740 DATA 32,112,120,120,252,252,254,255
1750 DATA 31,31,31,63,63,31,63,63
1760 DATA 255,255,255,255,254,252,248,224
1770 DATA 240,224,128,0,0,0,0,0
1780 DATA 255,127,127,63,63,63,31,31
1790 DATA 255,255,255,255,255,255,255,252
1800 DATA 193,129,195,195,129,0,0,0
1810 DATA 224,224,192,128,0,0,0,0
1820 DATA 7,15,31,31,15,15,7,7
1830 DATA 128,192,192,224,224,240,240,240
1840 DATA 63,63,127,127,63,31,63,63
1850 DATA 255,255,255,255,254,255,252,248
1860 DATA 224,192,128,0,0,0,0,0
1870 DATA 63,63,31,7,3,0,0,0
1880 DATA 252,248,248,240,192,0,0,0
1890 DATA 7,7,3,3,0,0,0,0
1900 DATA 255,255,255,238,128,0,0,0
```

```
1910 DATA 255,255,159,31,15,15,7,3
1920 DATA 240,240,224,192,192,128,0,0
1930 DATA 0,0,0,1,1,3,3,7
1940 DATA 127,127,255,255,255,254,252,252
1950 DATA 240,240,192,128,0,0,0,0
1960 DATA 252,0,124,124,28,28,0,0
1970 DATA 7,3,3,1,0,0,0,0
1980 DATA 252,248,248,240,96,0,0,0
1990 SYMBOL AFTER 65
1995 DIM A(8)
2000  GOSUB 9000
2010 LOCATE 6,1:PRINT"ABCDE";:FOR I=1 TO 4:PRINT C
HR$(143);:NEXT:PRINT"F"
2020 LOCATE 4,2:PRINT"GHIJK L";:FOR I=1 TO 4:PRINT
 CHR$(143);:NEXT:PRINT"M":LOCATE 27,2:PRINT"N"
2030 LOCATE 5,3:PRINT"O PQ RS";:FOR I=1 TO 3:PRINT
 CHR$(143);:NEXT:PRINT"T":LOCATE 26,3:PRINT"UVW"
2040 LOCATE 5,4:PRINT"XYZ"
2050  GOSUB 9000
2060 LOCATE 9,4:PRINT"AB C";CHR$(143);"DE": LOCATE
 24,4:PRINT"FG";CHR$(143);CHR$(143);"HIJKL"
2070 LOCATE 1,5:PRINT"MNOPQRSTUVWX";CHR$(143);"YZ"
2080  GOSUB 9000
2090 LOCATE 19,5:PRINT"ABC D";:FOR I=1 TO 9:PRINT
CHR$(143);:NEXT
2100 LOCATE 1,6:PRINT"E";:FOR I=1 TO 6:PRINT CHR$(
143);:NEXT:PRINT"FGHIJKLM  NOPQR";:FOR I=1 TO 10:P
RINT CHR$(143);:NEXT
2110 LOCATE 1,7:PRINT"STU";CHR$(143);CHR$(143);CHR
$(143);"V WX YZ"
2120  GOSUB 9000
2130 LOCATE 18,7:PRINT"AB";:FOR I=1 TO 11:PRINT CH
R$(143);:NEXT:PRINT"CD"
2140 LOCATE 3,8:PRINT"EF";CHR$(143);CHR$(143);CHR$
(143);"GH";CHR$(143);"I    JKLM";:FOR I=1 TO 11:PR
INT CHR$(143);:NEXT:PRINT"N"
2150 LOCATE 4,9:PRINT"O";:FOR I=1 TO 6:PRINT CHR$(
143);:NEXT:PRINT"P    QRS";:FOR I=1 TO 12:PRINT CH
R$(143);:NEXT:PRINT"T"
2160 LOCATE 5,10:PRINT"U";:FOR I=1 TO 4:PRINT CHR$
(143);:NEXT:PRINT"VW    XYZ"
2170 GOSUB 9000
2180 LOCATE 19,10:PRINT"ABC";:FOR I=1 TO 8:PRINT C
HR$(143);:NEXT:PRINT"D"
2190 LOCATE 5,11:PRINT"E";CHR$(143);CHR$(143);CHR$
(143);"F      GHIJK";:FOR I=1 TO 8:PRINT CHR$(143)
;:NEXT:PRINT"L"
2200 LOCATE 6,12:PRINT"MNOP      Q";:FOR I=1 TO 5:
PRINT CHR$(143);:NEXT:PRINT"RST";CHR$(143);CHR$(14
3);CHR$(143);"U"
```

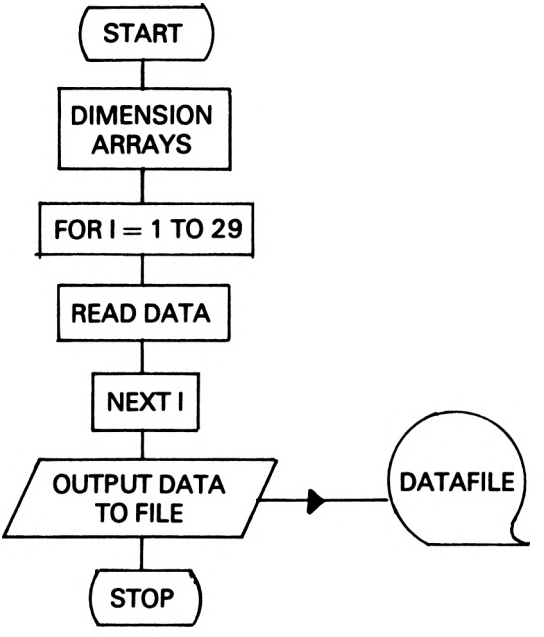
```
2210 LOCATE 6,13:PRINT"VWXYZ"
2220 GOSUB 9000
2230 LOCATE 11,13:PRINT"A    B";:FOR I=1 TO 4:PRIN
T CHR$(143);:NEXT:PRINT"CDEFGHIJ"
2240 LOCATE 9,14:PRINT"K";CHR$(143);"LM   NO";:FOR
 I=1 TO 4:PRINT CHR$(143);:NEXT:PRINT"P QR S"
2250 LOCATE 9,15:PRINT"T";CHR$(143);CHR$(143);"UV
   W";CHR$(143);CHR$(143);"X     Y"
2260 LOCATE 9,16:PRINT"Z";CHR$(143);CHR$(143);CHR$
(143)
2270 GOSUB 9000
2280 LOCATE 13,16:PRINT"A    B";CHR$(143);CHR$(143
);"CD     EFGH"
2290 LOCATE 10,17:PRINT"I";CHR$(143);"JK     LMNO
    P";CHR$(143);CHR$(143);CHR$(143);"Q"
2300 LOCATE 10,18:PRINT"RST      UV       WXY";CHR
$(143);"Z"
2310 LET F=70
2320 GOSUB 9000
2330 LOCATE 9,19:PRINT"ABC":LOCATE 31,19:PRINT"D":
LOCATE 9,20:PRINT"EF"
2990 SPEED WRITE 1
3000 SAVE "!WORLD",B,&C000,16383
8999 STOP
9000 FOR I=65 TO F
9010 FOR J=1 TO 8
9020 READ A(J)
9030 NEXT J
9040 SYMBOL I,A(1),A(2),A(3),A(4),A(5),A(6),A(7),A
(8)
9050 NEXT I
9060 RETURN
```

Create

This final program is used to initialise the database used for the quiz. A number of towns, cities and countries, together with their locations, are read from data and then transferred on to Tape B.

Flowchart

This flowchart represents the general operation of the program.



Major variables

The following table shows the main variables used in the program.

N$()	Name of place
PX()	X co-ordinate of location
PY()	Y co-ordinate of location
TY$()	Code for map: B=Britain W=World

Comments on the program

Create is a very short program which operates as follows:

Lines 10-20
Clear screen and dimension arrays.

Lines 30-50
Read information from data statements.

Lines 60-130
Output database from memory on to tape.

Lines 140-420
Data containing information on both maps.

The listing

Enter the program carefully and save the database on to tape as indicated at the beginning of the chapter. It is not necessary to save the actual program.

```
10 CLS
20 DIM N$(30),PX(30),PY(30),TY$(30)
30 FOR I=1 TO 29
40 READ N$(I),PX(I),PY(I),TY$(I)
50 NEXT I
60 OPENOUT "RECORDS"
70 FOR I=1 TO 29
80 PRINT#9,N$(I)
90 PRINT#9,PX(I)
100 PRINT#9,PY(I)
110 PRINT#9,TY$(I)
120 NEXT I
130 CLOSEOUT
140 DATA LONDON,304,115,B
150 DATA BRISTOL,221,124,B
160 DATA PORTSMOUTH,246,97,B
170 DATA EASTBOURNE,300,89,B
180 DATA MARGATE,326,110,B
190 DATA ANGLESEY,209,200,B
200 DATA LIVERPOOL,239,199,B
```

```
210 DATA ABERDEEN,289,330,B
220 DATA BRIGHTON,280,88,B
230 DATA LONDON,263,272,W
240 DATA NEW YORK,165,264,W
250 DATA PANAMA,131,195,W
260 DATA PARIS,265,259,W
270 DATA MADRID,252,245,W
280 DATA OSLO,277,287,W
290 DATA ROME,282,240,W
300 DATA DAKAR,243,196,W
310 DATA NEW ORLEANS,118,223,W
320 DATA CALCUTTA,396,204,W
330 DATA BOMBAY,378,204,W
340 DATA BREST,256,260,W
350 DATA NORWICH,332,156,B
360 DATA GLASGOW,235,284,B
370 DATA PLYMOUTH,186,97,B
380 DATA CORK,132,177,B
390 DATA DUBLIN,170,206,B
400 DATA GALWAY,97,233,B
410 DATA CARDIFF,209,132,B
420 DATA PRESTATYN,221,201,B
```

12. Adders and Ladders

This is the familiar game of Snakes and Ladders, with the additional twist that each player must solve a mathematical problem if he is to earn the right to throw the die. The aim of the game is to make your way from the bottom of the board to the top, trying to climb the ladders but avoiding the snakes which you must slide down.

Instructions

The game can be played by up to four players, who proceed around the board by rolling a die. The number on the die dictates how many squares you can move. To move your counter into the home at the end of the game, the die must show the exact number of squares you need to move.

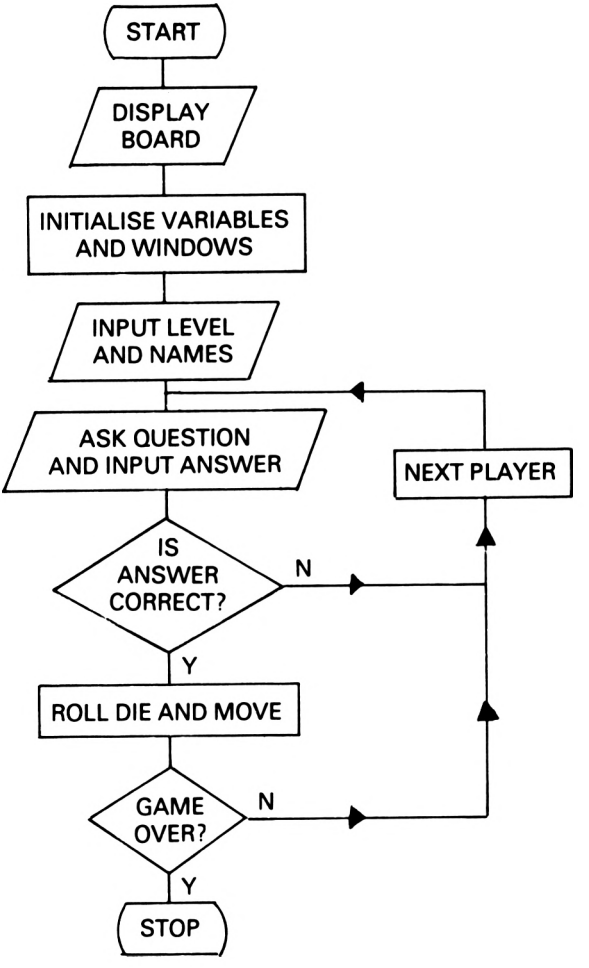
When the program is executed, the board, the die and two windows for messages are shown on the screen, together with the message 'PLEASE WAIT', which is displayed while the computer gives initial values to the many variables involved. When this process is complete you are asked to input a level of play from 1-5. This determines the difficulty of the mathematical questions which you will be asked during the game, with 1 being the easiest and 5 the most difficult. After selecting this you are asked how many players will be playing the game. Your response to this should be 1, 2, 3 or 4, since the game permits a maximum of four players. Next, the players' names are input (use four letters only), and these appear beside the counter which each player is going to use.

Then the game commences, and the first player is given a sum to answer. There is no time limit, but the object of the game is mental arithmetic so the use of pen and paper (or even a calculator) is not recommended unless you are playing on a high level of difficulty. If your answer is correct then you proceed to roll the die and your counter will be moved by the appropriate amount. Should your answer be incorrect, the correct answer is displayed on the screen and no movement occurs.

The program

Flowchart

The flowchart shows 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.

NAME$()	The players' names
PS()	The position of the counters
SQU()	The contents of the square which the counter is currently occupying
B()	The last piece to arrive at a certain board location
N	The player who is currently under consideration
NP$	The character signifying the number of players
NP	The number of players
ROLL	The number on the die
LEV$	The character signifying the level
LEV	The numerical equivalent of LEV$
A,B,C	The three numbers in the sum
D	This signifies the operation
SUM$	The question
T	The correct solution
ANS$	The character representing the player's answer
ANS	The numerical equivalent of ANS$
EL	End of ladder
ES	End of snake
X,Y	The co-ordinates of the top left-hand corner of the counter

Comments on the program

The program is very structured, and the following is an account of the operation of the particular routines.

Lines 10-80
This section dimensions the arrays, selects MODE 0 to allow as many colours as possible and then calls many of the later routines to display the board and windows.

Lines 90-130
While the message 'PLEASE WAIT' is shown the contents of

SQU() and PS() are given their initial values.

Line 140
This line calls the routine which asks for the level of the play and the players' names.

Lines 150-430
This is the main program which addresses the relevant routines to ask the question, roll the die and move the counters around the board.

Lines 440-630
The initial routine which uses the routines to draw the snakes and ladders and then complete the board.

Lines 640-1180
The section of program which draws the five snakes on the screen.

Lines 1190-1350
The corresponding section of code which produces the three ladders.

Lines 1360-1390
This routine draws the outline of the die in the top right-hand corner of the screen.

Lines 1400-1470
This routine is used to roll the die and obtain a random integer in the range 1-6. This code then creates the image of the die by plotting points in the appropriate place on the face of the die drawn in the above routine.

Lines 1480-1510
The right-hand instruction window is created, cleared and outlined.

Lines 1520-1580
The problem window is only one line high, so to give it the appearance of being larger than this some blank characters are drawn around the outside.

Lines 1590-1910
The routine to obtain the level of play, the number of players and their names, and display them on the screen beside their adopted counters.

Lines 1920-1990
This section of code computes the values of X and Y, which are the co-ordinates of the position where the counter should be drawn on the screen.

Lines 2000-2060
This short routine remembers the original contents of a square before it is occupied by a counter.

Lines 2070-2130
A related routine which draws back the original contents when a counter is moved from a square.

Lines 2140-2200
This routine is used to perform the task of lines 2000-2060 when there is more than one counter on any square.

Lines 2210-2260
A section of program to display a message when a player throws the die and obtains a number which will take him off the board at the end of the game, since the exact number of squares to be moved must be obtained.

Lines 2270-2480
The random mathematical problem is generated here and displayed on the screen. The program then takes the appropriate action of displaying the correct solution or returning to the main program to move the counter, depending on your answer.

Lines 2490-2600
This routine is used when you have the misfortune to land on the head of a snake. The snake is flashed and your counter is repositioned at the snake's tail.

Lines 2610-2700
A similar section of code to replace your counter at the top of a ladder when you land at its base.

Lines 2710-2820
The end game routine which plays the short jingle and displays the 'game over' message.

Lines 2830-2970
This is where the user-defined graphics are given their values. These are used for the counters and in the construction of the snakes.

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 saved in the caps lock mode. All spaces within the text should be entered as they appear.

```
10 DIM NAME$(4),PS(4),SQU(5,16,8),B(63)
20 MODE 0
30 GOSUB 2830
40 GOSUB 440
50 GOSUB 1360
60 GOSUB 1400
70 GOSUB 1480
80 GOSUB 1520
90 PEN #2,14:PRINT #2,"   PLEASE WAIT";:PEN #2,2
100 FOR N=1 TO 4
110 LET PS(N)=1
120 LET X=8:LET Y=90:GOSUB 2000
130 NEXT N
140 GOSUB 1590
150 FOR N=1 TO NP
160 GOSUB 1920:PLOT 640,400,1:MOVE X,Y:TAG:PRINT C
HR$(243+N);:TAGOFF
170 GOSUB 2270
180 IF T<>ANS THEN GOTO 390
190 LET Z$=INKEY$:IF Z$<>" " THEN GOTO 190
200 GOSUB 1400
210 IF PS(N)+ROLL>63 THEN GOSUB 2210:GOTO 390
220 LET TT=0:FOR K=1 TO NP
230 IF PS(K)=PS(N) THEN LET TT=TT+1
240 NEXT K
250 IF TT=1 OR B(PS(N))=N OR PS(N)=1 THEN GOSUB 19
20:GOSUB 2070
260 LET N1=N:FOR N=NP TO 1 STEP -1
270 IF N=N1 THEN GOTO 290
280 GOSUB 1920:PLOT 640,400,1:MOVE X,Y:TAG:PRINT C
```

```
HR$(243+N);:TAGOFF
290 NEXT N
300 LET N=N1:IF B(PS(N))=N THEN LET B(PS(N))=0
310 LET PS(N)=PS(N)+ROLL
320 IF B(PS(N))<>0 THEN GOSUB 2140:GOTO 340
330 IF B(PS(N))=0 THEN GOSUB 1920:GOSUB 2000
340 LET B(PS(N))=N
350 IF PS(N)=63 THEN PLOT 640,400,1:MOVE 32,384:TA
G:PRINT CHR$(243+N);:TAGOFF:GOTO 2710
360 GOSUB 1920:PLOT 640,400,1:MOVE X,Y:TAG:PRINT C
HR$(243+N);:TAGOFF
370 IF PS(N)=22 OR PS(N)=34 OR PS(N)=51 OR PS(N)=5
9 OR PS(N)=61 THEN GOTO 2490
380 IF PS(N)=7 OR PS(N)=35 OR PS(N)=42 THEN GOTO 2
610
390 FOR Z=1 TO 1000:NEXT Z
400 NEXT N
410 GOTO 150
420 IF INKEY$="" THEN GOTO 420
430 MODE 1:END
440 REM ** INITIAL ROUTINE **
450 INK 0,20:INK 1,0:INK 2,6:INK 3,2:INK 4,18:INK
5,26:INK 6,25:INK 7,15:INK 8,8:INK 10,6:INK 11,6:I
NK 12,6:INK 14,6,26:INK 15,26
460 BORDER 2:PAPER 3:CLS
470 WINDOW #1,1,12,1,21:PAPER #1,0:CLS #1
480 GOSUB 640
490 PLOT 4,384,2:TAG:PRINT "END";:TAGOFF
500 PLOT 640,400,3:FOR I=1 TO 4
510 IF I<>4 THEN MOVE 8,70+84*I:TAG:PRINT CHR$(240
);:TAGOFF:MOVE 8,54+84*I:TAG:PRINT CHR$(147);:TAGO
FF
520 MOVE 348,28+84*I:TAG:PRINT CHR$(240);:TAGOFF:M
OVE 348,12+84*I:TAG:PRINT CHR$(153);:TAGOFF
530 NEXT I
540 FOR I=0 TO 384 STEP 48
550 IF I=48 THEN PLOT I,62:DRAW I,356:GOTO 570
560 PLOT I,62,1:DRAW I,398
570 NEXT I
580 FOR I=62 TO 398 STEP 42
590 PLOT O,I,1:DRAW 384,I
600 NEXT I
610 GOSUB 1190
620 PLOT 640,400,2:MOVE 8,90:TAG:PRINT CHR$(243);:
TAGOFF
630 RETURN
640 REM ** SNAKES **
650 PLOT 640,400,8
660 MOVE 246,372:TAG:PRINT CHR$(249);
670 MOVE 230,356:PRINT CHR$(254);
```

```
680 MOVE 222,340:PRINT CHR$(254);
690 MOVE 222,324:PRINT CHR$(255);
700 MOVE 238,308:PRINT CHR$(251);
710 MOVE 238,292:PRINT CHR$(255);
720 MOVE 254,276:PRINT CHR$(251);
730 MOVE 270,260:PRINT CHR$(251);
740 MOVE 262,244:PRINT CHR$(254);
750 MOVE 256,228:PRINT CHR$(250);
760 MOVE 240,214:PRINT CHR$(250);
770 MOVE 224,198:PRINT CHR$(250);
780 MOVE 210,182:PRINT CHR$(248);
790 PLOT 640,400,4
800 MOVE 246,166:PRINT CHR$(249);
810 MOVE 230,150:PRINT CHR$(250);
820 MOVE 210,134:PRINT CHR$(250);
830 MOVE 190,118:PRINT CHR$(250);
840 MOVE 170,102:PRINT CHR$(250);
850 MOVE 156,86:PRINT CHR$(248);
860 PLOT 640,400,5
870 MOVE 102,330:PRINT CHR$(249);
880 MOVE 102,314:PRINT CHR$(251);
890 MOVE 118,298:PRINT CHR$(251);
900 MOVE 134,282:PRINT CHR$(251);
910 MOVE 150,266:PRINT CHR$(251);
920 MOVE 150,250:PRINT CHR$(248);
930 PLOT 640,400,7
940 MOVE 54,246:PRINT CHR$(249);
950 MOVE 38,230:PRINT CHR$(255);
960 MOVE 46,214:PRINT CHR$(255);
970 MOVE 54,198:PRINT CHR$(255);
980 MOVE 50,182:PRINT CHR$(254);
990 MOVE 60,166:PRINT CHR$(251);
1000 MOVE 68,150:PRINT CHR$(251);
1010 MOVE 84,134:PRINT CHR$(131);
1020 MOVE 96,150:PRINT CHR$(254);
1030 MOVE 104,166:PRINT CHR$(254);
1040 MOVE 120,182:PRINT CHR$(250);
1050 MOVE 150,182:PRINT CHR$(251);
1060 MOVE 160,166:PRINT CHR$(208);
1070 PLOT 640,400,6
1080 MOVE 150,372:PRINT CHR$(249);
1090 MOVE 150,356:PRINT CHR$(251);
1100 MOVE 154,340:PRINT CHR$(255);
1110 MOVE 160,324:PRINT CHR$(255);
1120 MOVE 160,308:PRINT CHR$(248);
1130 PLOT 278,366,1
1140 PLOT 278,160,1
1150 PLOT 134,324,1
1160 PLOT 86,240,1
1170 PLOT 182,366,1
```

```
1180 RETURN
1190 REM ** LADDERS **
1200 PLOT 300,282,10:DRAWR 0,100
1210 PLOT 324,282:DRAWR 0,100
1220 FOR J=1 TO 5
1230 PLOT 300,276+18*J:DRAWR 24,0
1240 NEXT J
1250 PLOT 300,72,11:DRAWR -146,142
1260 PLOT 324,72:DRAWR -146,142
1270 FOR J=2 TO 9
1280 PLOT 325-17.5*J,50+17*J:DRAWR 20,0
1290 NEXT J
1300 PLOT 108,240,12:DRAWR 48,100
1310 PLOT 132,240:DRAWR 48,100
1320 FOR J=1 TO 5
1330 PLOT 102+10*J,236+18*J:DRAWR 24,0
1340 NEXT J
1350 RETURN
1360 REM ** DIE **
1370 WINDOW #1,16,17,2,4:PAPER #1,15:PEN #1,1:CLS
#1
1380 PLOT 478,384,1:DRAWR 68,0:DRAWR 0,-50:DRAWR -
68,0:DRAWR 0,50
1390 RETURN
1400 REM ** ROLL DIE **
1410 CLS #1
1420 LET ROLL=1+INT(RND(1)*6)
1430 IF ROLL=1 OR ROLL=3 OR ROLL=5 THEN PLOT 512,3
58,1
1440 IF ROLL>1 THEN PLOT 500,372,1:PLOT 524,344
1450 IF ROLL>3 THEN PLOT 500,344,1:PLOT 524,372
1460 IF ROLL=6 THEN PLOT 500,358,1:PLOT 524,358,1
1470 RETURN
1480 REM ** INSTRUCTIONS WINDOW **
1490 WINDOW #0,14,19,6,21:PAPER #0,15:PEN #0,2:CLS

1500 PLOT 414,320,1:DRAWR 194,0:DRAWR 0,-258:DRAWR
 -194,0:DRAWR 0,258
1510 RETURN
1520 REM ** PROBLEM WINDOW **
1530 PLOT 16,40,15:TAG:PRINT STRING$(18,CHR$(143))
;
1540 PLOT 16,24,15:TAG:PRINT STRING$(18,CHR$(143))
;
1550 TAGOFF
1560 PLOT 16,40,1:DRAWR 576,0:DRAWR 0,-32:DRAWR -5
76,0:DRAWR 0,32
1570 WINDOW #2,2,18,24,24:PAPER #2,15:CLS #2:PEN #
2,2
1580 RETURN
```

```
1590 REM ** INPUT PLAYERS **
1600 CLS #2
1610 PRINT #2,"INPUT LEVEL (1-5)";
1620 LET LEV$=INKEY$:IF LEV$="" THEN GOTO 1620
1630 IF LEV$>"0" AND LEV$<"6" THEN GOTO 1650
1640 GOTO 1620
1650 LET LEV=VAL(LEV$)
1660 CLS #2
1670 PRINT #2,"HOW MANY PLAYERS?";
1680 LET NP$=INKEY$:IF NP$="" THEN GOTO 1680
1690 IF NP$>"0" AND NP$<"5" THEN GOTO 1710
1700 GOTO 1680
1710 CLS #2
1720 LET NP=VAL(NP$)
1730 CLS :PRINT:PRINT "NAMES:"
1740 FOR I=1 TO NP
1750 PEN 3:PRINT CHR$(243+I);" ";:PEN 2
1760 FOR J=1 TO 4
1770 LET Z$=INKEY$:IF Z$="" THEN GOTO 1770
1780 IF ASC(Z$)=13 THEN GOTO 1850
1790 IF Z$>="A" AND Z$<="Z" THEN GOTO 1810
1800 GOTO 1770
1810 LET NAME$(I)=NAME$(I)+Z$
1820 PRINT Z$;
1830 NEXT J
1840 GOTO 1890
1850 FOR K=J TO 4
1860 LET NAME$(I)=NAME$(I)+" "
1870 PRINT " ";
1880 NEXT K
1890 NEXT I
1900 PEN 1
1910 RETURN
1920 REM ** LOCATE POSITION **
1930 LET X=-40:LET Y=90:LET Z=PS(N):LET Z1=0
1940 IF Z<=8 THEN GOTO 1970
1950 LET Y=Y+42:LET Z=Z-8:LET Z1=Z1+1
1960 GOTO 1940
1970 IF INT(Z1/2)<>Z1/2 THEN X=X+48*(9-Z)
1980 IF INT(Z1/2)=Z1/2 THEN X=X+48*Z
1990 RETURN
2000 REM ** REMEMBER THE CONTENTS **
2010 FOR Q=1 TO 16
2020 FOR R=1 TO 8
2030 LET SQU(N,Q,R)=TEST(X+2*(Q-1),Y-2*(R-1))
2040 NEXT R
2050 NEXT Q
2060 RETURN
2070 REM ** DRAW OLD SQUARE **
2080 FOR Q=1 TO 16
```

```
2090 FOR R=1 TO 8
2100 PLOT X+2*(Q-1),Y-2*(R-1),SQU(N,Q,R)
2110 NEXT R
2120 NEXT Q
2130 RETURN
2140 REM ** OVERDRAW **
2150 FOR Q=1 TO 16
2160 FOR R=1 TO 8
2170 LET SQU(N,Q,R)=SQU(B(PS(N)),Q,R)
2180 NEXT R
2190 NEXT Q
2200 RETURN
2210 REM ** OFF BOARD **
2220 CLS #2:PRINT #2,"    TOO LARGE";
2230 SOUND 1,127,50:SOUND 1,134,50:SOUND 1,142,50:
SOUND 1,150,100
2240 FOR Z=1 TO 2000:NEXT Z
2250 CLS #2
2260 RETURN
2270 REM ** PROBLEM **
2280 LOCATE 1,10:PRINT SPACE$(36);
2290 PEN #2,2:PRINT #2,NAME$(N);:PEN #2,3
2300 LET A=1+INT(RND(1)*6*LEV)
2310 LET D=1+INT(RND(1)*2)
2320 LET B=4+INT(RND(1)*6)
2330 IF D=1 THEN LET SUM$=STR$(A)+" +"+STR$(B):LET
 T=A+B
2340 IF D=2 THEN LET SUM$=STR$(A)+" x"+STR$(B):LET
 T=A*B
2350 LET D=1+INT(RND(1)*2)
2360 LET C=1+INT(RND(1)*4*LEV)
2370 IF D=1 THEN LET SUM$=SUM$+" +"+STR$(C):LET T=
T+C
2380 IF D=2 THEN LET SUM$=SUM$+" -"+STR$(C):LET T=
T-C
2390 PEN #2,1:PRINT #2,SUM$;:PEN #2,2
2400 FOR Z=1 TO 5000:NEXT Z
2410 LOCATE 1,10:PRINT NAME$(N);"TYPE  IN    YOUR
 ANSWER      ";
2420 PEN 2
2430 LOCATE 2,15:LINE INPUT ANS$:LET ANS=VAL(ANS$)
2440 PEN 1
2450 IF T<>ANS THEN SOUND 1,1276,100:SOUND 1,1607,
100:LOCATE 1,10:PRINT "ANSWERIS    ";:PEN 2:PRINT
USING "####";T:PEN 1:PRINT SPACE$(18):FOR Z=1 TO 2
000:NEXT Z:RETURN
2460 SOUND 1,119,20:SOUND 1,106,20:SOUND 1,95,20:S
OUND 1,89,50
2470 LOCATE 1,10:PRINT"TYPE  SPACE TO    ROLL  THE
   DIE   ";
```

```
2480 RETURN
2490 REM ** DOWN SNAKE **
2500 IF PS(N)=22 THEN ES=4:INK 4,18,20
2510 IF PS(N)=34 THEN ES=20:INK 7,15,20
2520 IF PS(N)=51 THEN ES=36:INK 5,26,20
2530 IF PS(N)=59 THEN ES=21:INK 8,8,20
2540 IF PS(N)=61 THEN ES=45:INK 6,25,20
2550 FOR P=1 TO 800 :SOUND 1,P,1:NEXT P
2560 INK 4,18:INK 5,26:INK 6,25:INK 7,15:INK 8,8
2570 GOSUB 1920:LET X1=X:LET Y1=Y:GOSUB 2070
2580 LET B(PS(N))=0
2590 LET PS(N)=ES
2600 GOTO 320
2610 REM ** UP LADDER **
2620 IF PS(N)=7 THEN LET EL=29:INK 11,6,20
2630 IF PS(N)=35 THEN LET EL=52:INK 12,6,20
2640 IF PS(N)=42 THEN LET EL=58:INK 10,6,20
2650 FOR P= 800 TO 1 STEP -1:SOUND 1,P,1:NEXT P
2660 INK 10,6:INK 11,6:INK 12,6
2670 GOSUB 1920:LET X1=X:LET Y1=Y:GOSUB 2070
2680 LET B(PS(N))=0
2690 LET PS(N)=EL
2700 GOTO 320
2710 REM ** GAME OVER **
2720 CLS #2:PEN #2,14
2730 PRINT #2,"G A M E   O V E R";
2740 RESTORE
2750 FOR I=1 TO 21
2760 READ A,B,C
2770 SOUND 1,A,10:SOUND 2,B,10:SOUND 4,C,10
2780 NEXT I
2790 IF INKEY$="" THEN GOTO 2790
2800 MODE 1:END
2810 DATA 119,0,0,119,0,0,119,0,0,159,0,0,169,0,0,
159,0,0,142,0,0,142,0,0,142,0,0,159,0,0,159,0,0,15
9,0,0,0,0,0,0,0,0,0,0,0
2820 DATA 127,179,638,127,179,638,127,179,638,119,
190,478,119,190,478,119,190,478
2830 REM ** USER DEFINED GRAPHICS **
2840 SYMBOL AFTER 244
2850 SYMBOL 244,&3C,&66,&C7,&E7,&E7,&C3,&7E,&3C
2860 SYMBOL 245,&3C,&66,&D3,&F7,&EF,&C3,&7E,&3C
2870 SYMBOL 246,&3C,&46,&FB,&E7,&FB,&DB,&66,&3C
2880 SYMBOL 247,&3C,&66,&D7,&D7,&D7,&C3,&76,&3C
2890 SYMBOL 248,&F,&F,&E,&1E,&F8,0,0,0
2900 SYMBOL 249,&38,&3E,&77,&7F,&7F,&7E,&F8,&F0
2910 SYMBOL 250,&F,&F,&1E,&1E,&3C,&3C,&78,&78
2920 SYMBOL 251,&F0,&F0,&78,&78,&3C,&3C,&1E,&1E
2930 SYMBOL 252,&F,&F,&1E,&1E,&3C,&3C,&1E,&1E
2940 SYMBOL 253,&F0,&F0,&78,&78,&3C,&3C,&78,&78
```

```
2950 SYMBOL 254,&F,&F,&F,&F,&1E,&1E,&1E,&1E
2960 SYMBOL 255,&1E,&1E,&1E,&1E,&F,&F,&F,&F
2970 RETURN
```

13. Blockbusters

You may have seen the popular television quiz for teenagers on which this game is based. In it, two contestants try to answer general knowledge questions, attempting to cross the board from one side to the other.

Instructions

The board on which our game is played contains twenty-five squares in the configuration of a five by five grid, as shown below:

```
A F B T C
Z C WV L
R S G K X
R E P J I
M O N H U
```

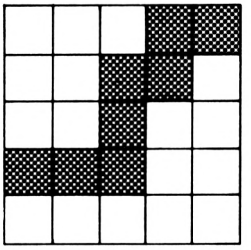
In each of the squares is a different letter, which is the initial letter of the answer to a question which is posed when a player selects that square. When the game is played the first person to press his button has the right to answer, and if he answers the question correctly the square is filled with his colour. If he is wrong the second player has an attempt, but if neither player answers correctly, a square must be reselected.

The game ends when one player has a complete line across the board or all the squares are occupied.

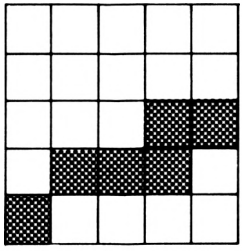
The first requirement on executing this program is to enter the database, which takes the usual amount of time. When the internal buffering has finished, the blockbuster board is displayed. The names of the two players are entered, with player one trying to connect left to right with yellow squares and player two working from top to bottom in blue. After these directions are given a coin is thrown inside the computer's memory to decide who is to have the initial choice of square. When this has been selected the game begins.

You are warned that the question is imminent by being told to place your fingers on your buzzers. Player one's buzzer is the 'KEY-1' on the top row of the keyboard; player two's is the 'KEY-2' of the numeric keypad on the right-hand side of the computer. When the question is displayed the first player to press his buzzer wins the right to answer the question (when the buzzer is pressed it should not be held down, as this can cause problems when typing in the answer).

The person who types in the correct answer within the time limit will shade the square in his colour. For a player to win he must create a pathway from one side of the board to the other, but diagonal steps are not permitted. Hence for player one, the path shown on the right below would not be a winning line, but the one on the left would.



good

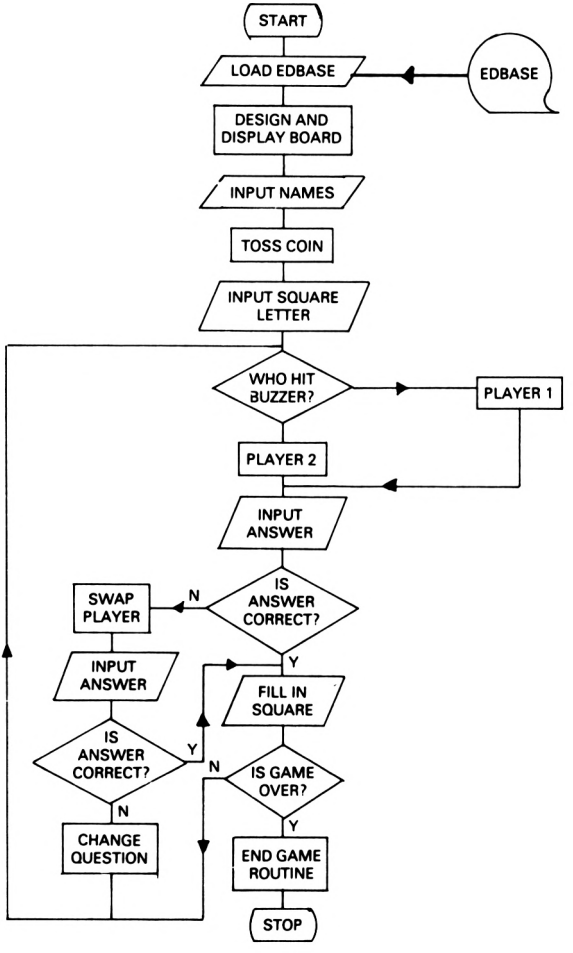


bad

The program

Flowchart

The flowchart on p.164 represents the general operation of the program.


START
LOAD EDBASE
EDBASE
DESIGN AND DISPLAY BOARD
INPUT NAMES
TOSS COIN
INPUT SQUARE LETTER
WHO HIT BUZZER?
PLAYER 1
PLAYER 2
INPUT ANSWER
IS ANSWER CORRECT?
N
SWAP PLAYER
Y
INPUT ANSWER
FILL IN SQUARE
IS ANSWER CORRECT?
Y
N
IS GAME OVER?
N
Y
CHANGE QUESTION
END GAME ROUTINE
STOP

Variables

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

W(x,y)	Question numbers in database
C(x,y)	Copy of W(x,y)
A$(x)	Words in the database
B$(x)	Clues in the database
I	Number of items in the database
NUM	Count of questions answered
J,K & L,M	x & y co-ordinates
TG & PG	Number of player having turn
V() & B()	Used in determining whether the game has been won
LD	Last position
N1,N2,N3	Notes used in the musical jingle
DV	Length of each note

Comments on the program

The program makes full use of subroutines, which are clearly marked in the listing and described below.

Lines 10-60
The arrays to be used are dimensioned and the initial ink colours are given their values.

Lines 70-240
This is the section of code which transfers the data from EDBASE into the computer's memory.

Lines 250-300
These lines select the questions which are to be asked during the quiz program. No two squares will contain the same initial letter, so this routine takes a variable amount of time to complete, but usually less than thirty seconds.

Lines 310-480
This is the routine which is used to display the grid on the screen with the initial letters in each of the twenty-five squares.

Lines 490-680
Here the players' names are entered, after which the computer will recap on the instructions and then simulate the throwing of a coin to decide who is to have the first move. This is a simple process which is often needed, since the first player in a game often has an advantage. Since there is an equal chance of a coin falling 'heads' or 'tails', this can be re-created by seeing if a random number in the range 0-1 is greater than or less than 0.5. For example,

```
IF RND(1)<0.5 THEN LET P=1 ELSE LET P=2
```

Lines 690-1080
This is the main program which is repeated until the game is finished. It handles the prompts and answering of the questions including the use of the time limit.

Lines 1090-1250
If neither player is able to answer the question, then another question must be chosen with the initial letter of the answer the same.

Lines 1260-1390
This is the routine which colours the squares when a player has answered a question correctly.

Lines 1400-1780
These lines perform a check to see whether one of the players has won the game. This is the most complicated routine in the program and functions by treating the board as a maze and seeing if the computer can find its way across. If this is not possible then there is no winning line.

Lines 1790-1860
This routine is used when the board is in a stalement position.

Lines 1870-1890
A short routine used when no connections across the board are discovered.

Lines 1900-2060
This is the 'end game' routine which plays the short jingle and declares the winner.

Lines 2070-2120
This routine draws the outline and grid lines on the screen when the board is created.

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:INK 1,24:INK 0,1:PEN 1:PAPER 0:CLS
20 BORDER 1
30 LOCATE 1,11
40 DIM C(5,5),V(30),B(30)
50 DIM A$(1001),B$(1001),CS$(40,2)
60 DIM W(5,5)
70 LET I=1
80 CLS
90 PRINT:PRINT"      B L O C K B U S T E R S ! !"
100 LOCATE 1,6:PRINT "PLACE THE DATABASE  CASSETTE
   INTO  THEDATACORDER"
110 PRINT:PRINT:PRINT:PRINT
120 OPENIN "!EDBASE"
130 INPUT #9,A$(I)
140 IF A$(I)="XXX" THEN GOTO 180
150 INPUT #9,B$(I):IF B$(I)="*" THEN GOTO 130
160 LET I=I+1
170 GOTO 130
180 LET CLUES=1
190 IF EOF THEN GOTO 240
200 INPUT #9,CS$(CLUES,1)
210 INPUT #9,CS$(CLUES,2)
220 LET CLUES=CLUES+1
230 GOTO 190
240 CLOSEIN
250 FOR K=1 TO 5:FOR J=1 TO 5:W(J,K)=0:NEXT J,K
260 FOR K=1 TO 5:FOR J=1 TO 5
270 X=1+INT(RND*(I-1)):IF B$(X)="*" THEN GOTO 270
280 FOR L=1 TO 5:FOR M=1 TO 5:IF LEFT$(A$(W(L,M)),
1)=LEFT$(A$(X),1) THEN GOTO 270 ELSE NEXT M,L
290 W(K,J)=X
```

```
300 NEXT J,K
310 P1S=0:P2S=0
320 NUM=0
330 BORDER 13:INK 2,13:INK 0,1:PAPER 2:CLS
340 INK 1,24
350 WINDOW#4,15,25,21,21:PEN#4,3:PAPER#4,2:WINDOW#
3,1,14,12,12:PEN#3,3:PAPER#3,2:WINDOW#2,27,40,12,1
2:PEN#2,3:PAPER#2,2
360 LOCATE #2,1,1:PRINT#2,"SCORE#2-  0"
370 LOCATE #3,1,1:PRINT#3,"SCORE#1-  0"
380 LOCATE#4,1,1:PRINT#4,"LETTER - "
390 FOR K=14 TO 24:FOR J=8 TO 18
400 PAPER 0:LOCATE K,J:PRINT" ":NEXT J,K
410 PAPER 0:FOR K=14 TO 24:LOCATE K,7:PRINT " ";:L
OCATE K,19:PRINT " ";:NEXT K
420 PAPER 1:FOR J=8 TO 18:LOCATE 13,J:PRINT " ";:L
OCATE 25,J:PRINT " ";:NEXT J
430 PAPER 3:LOCATE 13,7:PRINT " ";:LOCATE 25,7:PRI
NT " ";:LOCATE 13,19:PRINT " ";:LOCATE 25,19:PRINT
 " ";:PAPER 0
440 GOSUB 2070
450 FOR K=1 TO 5
460 FOR J=1 TO 5
470 PLOT 200+(K*16)+(20*(K-1)),314-((J*70)/2),1:TA
G:PRINT LEFT$(A$(W(K,J)),1);:NEXT J,K:TAGOFF
480 PEN 1:LOCATE 1,1:PAPER 2:PRINT"      B L O C K
 B U S T E R S ! !"
490 PEN 3:PRINT:LINE INPUT "PLAYER 1 NAME - ";P1$

500 LOCATE 1,3:PRINT"
          ":LOCATE 1,3:LINE INPUT "PLAYER 2 NAME
- ";P2$
510 LOCATE 1,3:PRINT"
          ":LOCATE 1,3:PRINT P1$;" IS TRYING TO M
OVE ACROSS THE BOARD AND ";P2$;" DOWN IT."
520 FOR J=1 TO 2000:NEXT J
530 LOCATE 1,3:PRINT"

                                    "
540 LOCATE 1,3:PRINT P1$;" IS ";:PEN 1:PRINT"YELLO
W.":PEN 3:PRINT P2$;" IS ";:PEN 0:PRINT"BLUE."
550 PEN 3
560 FOR J=1 TO 2000:NEXT J
570 LOCATE 1,3:PRINT"
 "
580 LOCATE 1,3:PRINT P1$;", HEADS OR TAILS ?"
590 IF INKEY$="H" THEN Q$="H":GOTO 620
600 IF INKEY$="T" THEN Q$="T":GOTO 620
610 GOTO 590
620 IF RND<0.5 THEN R$="HEADS" ELSE R$="TAILS"
```

```
630 LOCATE 1,3:PRINT"
          ":LOCATE 1,3:PRINT"IT IS ";R$;"."
640 IF Q$<>LEFT$(R$,1) THEN PG=2:PRINT:PRINT P2$;"
 TO GO FIRST."
650 IF Q$=LEFT$(R$,1) THEN PG=1:PRINT:PRINT P1$;"
TO GO FIRST."
660 TG=PG
670 FOR J=1 TO 2000:NEXT J:LOCATE 1,3
680 PRINT"
                                    "
690 TG=PG
700 LOCATE #4,10,1:PRINT#4," "
710 PEN 3:LOCATE 1,3:IF TG=1 THEN PRINT P1$; ELSE
PRINT P2$;
720 PRINT", WHICH SQUARE ?"
730 Q$=INKEY$:IF Q$="" THEN GOTO 730
740 FOR K=1 TO 5:FOR J=1 TO 5
750 IF Q$=LEFT$(A$(W(J,K)),1) THEN GOTO 770
760 NEXT J,K:GOTO 730
770 LOCATE#4,10,1:PRINT#4,Q$
780 LOCATE 1,3:PRINT"
          "
790 LOCATE 1,3:PRINT"FINGERS ON THE BUZZERS!"
800 FOR B=1 TO 1000:NEXT B
810 PRINT:PRINT"THE CLUE IS - ":PEN 3
820 IF LEN(B$(W(J,K)))>2 THEN PEN 0:PRINT B$(W(J,K
)):GOTO 860
830 FOR A=1 TO CLUES
840 IF CS$(A,1)=B$(W(J,K)) THEN PEN 0:PRINT CS$(A,
2):GOTO 860
850 NEXT A
860 IF INKEY(64)=0 THEN TG=1:SOUND 1,100,100:GOTO
890
870 IF INKEY(14)=0 THEN TG=2:SOUND 1,200,100:GOTO
890
880 GOTO 860
890 LOCATE 1,3:PRINT"
                                    "
900 FOR B=1 TO 50:Q$=INKEY$:NEXT B
910 PEN 3:LOCATE 1,3:IF TG=1 THEN PRINT P1$; ELSE
PRINT P2$;
920 PRINT", THE ANSWER ? ";:GOSUB 1160:IF OOT=1 TH
EN LOCATE 1,3:PRINT"
          ":LOCATE 1,3:PRINT"OUT OF TIME.":GOTO 96
0
930 IF Q$=A$(W(J,K)) THEN PG=TG:GOTO 1260
940 LOCATE 1,3:PRINT"
```

```
                                        "
950 LOCATE 1,3:PRINT"SORRY,THAT IS WRONG."
960 FOR B=1 TO 1000:NEXT B:LOCATE 1,3:PRINT"
                                           "
970 IF TG=1 THEN TG=2 ELSE TG=1
980 PEN 3:LOCATE 1,3:IF TG=1 THEN PRINT P1$; ELSE
PRINT P2$;
990 PRINT", THE ANSWER ? ";:GOSUB 1160:IF OOT=1 TH
EN LOCATE 1,3:PRINT"
          ":LOCATE 1,3:PRINT"OUT OF TIME.":PRINT:P
RINT"IT WAS :-";A$(W(J,K)):GOTO 1030
1000 IF Q$=A$(W(J,K)) THEN PG=TG:GOTO 1260
1010 LOCATE 1,3:PRINT"
                                                  "
1020 LOCATE 1,3:PRINT"SORRY,THAT IS WRONG.":PRINT:
PRINT"IT WAS :- ";A$(W(J,K))
1030 GOSUB 1090
1040 FOR B=1 TO 2000:NEXT B:LOCATE 1,3:PRINT"



     "
1050 IF NC=1 THEN LET NC=0:GOTO 690
1060 IF PG=1 THEN PG=2 ELSE PG=1
1070 GOTO 690
1080 FOR B=1 TO 2500:NEXT B:GOTO 250
1090 REM **REPLACE QUESTION**
1100 LET NC=1
1110 X=1+INT(RND*(I-1)):IF B$(X)="*" THEN GOTO 111
0
1120 IF LEFT$(A$(X),1)=LEFT$(A$(W(J,K)),1) THEN W(
J,K)=X:GOTO 1140
1130 GOTO 1110
1140 PLOT 200+(J*16)+(20*(J-1)),314-((K*70)/2),1:T
AG:PRINT LEFT$(A$(W(J,K)),1);:TAGOFF
1150 RETURN
1160 OOT=0:FLG=0:AFTER 250 GOSUB 1190
1170 IF FLG=1 THEN RETURN ELSE Q1$=INKEY$:IF Q1$="
" THEN GOTO 1170
1180 GOTO 1210
1190 PRINT:OOT=1
1200 FLG=1:RETURN
1210 PRINT Q1$;
1220 IF FLG=1 THEN RETURN ELSE Q2$=INKEY$:IF Q2$="
" THEN GOTO 1220
1230 PRINT Q2$;:LINE INPUT Q3$
1240 Q$=Q1$+Q2$+Q3$
1250 RETURN
1260 REM **CORRECT ANSWER**
```

```
1270 NUM=NUM+1
1280 IF TG=1 THEN COL=1 ELSE COL=0
1290 FOR B=288-((K-1)*35)-35 TO 288-((K-1)*35)
1300 PLOT 207+((J-1)*36),B,COL:DRAWR 34,0:NEXT B
1310 GOSUB 2070
1320 W(J,K)=2000+TG
1330 IF TG=1 THEN P1S=P1S+10 ELSE P2S=P2S+10
1340 IF TG=1 THEN LOCATE#3,9,1:PRINT#3,USING "###"
;P1S ELSE LOCATE#2,9,1:PRINT#2,USING "###";P2S
1350 PAPER 2:LOCATE 1,3:PRINT"
                    "
1360 LOCATE 1,3:PRINT"THAT IS CORRECT."
1370 PG=TG
1380 FOR B=1 TO 2000:NEXT B
1390 LOCATE 1,3:PRINT"
          "
1400 REM **CHECK FOR WIN**
1410 FOR B=1 TO 5
1420 U(B)=1:D(B)=1:F(B)=1:NEXT B
1430 IF TG=2 THEN GOTO 1610
1440 FOR L=1 TO 5:FOR M=1 TO 5:IF W(L,M)=2001 THEN
 C(L,M)=1 ELSE C(L,M)=0
1450 NEXT M,L
1460 FOR L=1 TO 5:IF C(1,L)= 1 THEN GOTO 1470 ELSE
 J=1:GOTO 1580
1470 T=1
1480 M=L:J=1
1490 LD=0
1500 V(T)=J:B(T)=M
1510 IF J=5 THEN GOTO 1900
1520 IF C(J+1,M)=1 THEN LD=0:J=J+1:T=T+1:GOTO 1500
1530 IF M=1 THEN GOTO 1560
1540 IF C(J,M-1)=1 AND LD<>3 THEN LD=1:M=M-1:T=T+1
:GOTO 1500
1550 IF M=5 THEN GOTO 1570
1560 IF C(J,M+1)=1 AND LD<>1 THEN LD=3:M=M+1:T=T+1
:GOTO 1500
1570 IF J=5 THEN GOTO 1900
1580 IF T<1 OR J=1 THEN NEXT L
1590 IF L=6 THEN GOTO 1870
1600 LD=0:T=T-1:C(J,M)=0:J=V(T):M=B(T):GOTO 1500

1610 REM **WIN#2**
1620 FOR L=1 TO 5:FOR M=1 TO 5:IF W(L,M)=2002 THEN
 C(L,M)=1 ELSE C(L,M)=0
1630 NEXT M,L
1640 FOR L=1 TO 5:IF C(L,1)= 1 THEN GOTO 1650 ELSE
 M=1:GOTO 1760
1650 T=1
1660 J=L:M=1
```

```
1670 LD=0
1680 V(T)=J:B(T)=M
1690 IF M=5 THEN GOTO 1900
1700 IF C(J,M+1)=1 THEN LD=0:M=M+1:T=T+1:GOTO 1680

1710 IF J=1 THEN GOTO 1740
1720 IF C(J-1,M)=1 AND LD<>3 THEN LD=1:J=J-1:T=T+1
:GOTO 1680
1730 IF J=5 THEN GOTO 1750
1740 IF C(J+1,M)=1 AND LD<>1 THEN LD=3:J=J+1:T=T+1
:GOTO 1680
1750 IF M=5 THEN GOTO 1900
1760 IF T<1 OR M=1 THEN NEXT L
1770 IF L=6 THEN GOTO 1870
1780 LD=0:T=T-1:C(J,M)=0:J=V(T):M=B(T):GOTO 1680
1790 REM **STALEMATE**
1800 LOCATE 1,3:PRINT"


                                    "
1810 LOCATE 11,3:PRINT"IT´S A STALEMATE."
1820 LOCATE 11,4
1830 IF P1S>P2S THEN PRINT P1$;" HAS THE MOST POIN
TS."
1840 IF P2S>P1S THEN PRINT P2$;" HAS THE MOST POIN
TS."
1850 IF P1S=P2S THEN PRINT"IT IS A DRAW ON POINTS.
"
1860 GOTO 1940
1870 REM **NO CONNECTION**
1880 IF NUM=25 THEN GOTO 1790
1890 GOTO 690
1900 REM **WIN**
1910 LOCATE 1,3:PRINT"


                                    "
1920 IF TG=1 THEN LOCATE 11,3:PRINT P1$;" HAS WON!
!"
1930 IF TG=2 THEN LOCATE 11,3:PRINT P2$;" HAS WON!
!"
1940 RESTORE 2010
1950 LET DU=20
1960 FOR II=1 TO 14
1970 BORDER II
1980 READ N1,N2,N3
1990 SOUND 1,N1,DU:SOUND 2,N2,DU:SOUND 4,N3,DU
2000 NEXT II
2010 DATA 319,956,190,358,956,213,379,956,239,319,
956,190,319,956,190,179,638,159,179,638,159,190,95
```

```
6,119,190,956,119,190,956,119,0,0,0,1911,1911,1911
,1911,1911,1911,1911,1911,1911
2020 BORDER 12
2030 LOCATE 11,23:PRINT"ANOTHER GO  (Y/N)? "
2040 LET AG$=INKEY$:IF AG$="" THEN GOTO 2040
2050 IF AG$="Y" THEN GOTO 250
2060 GOTO 2060
2070 REM **DRAW GRID**
2080 INK 3,0:PLOT 207,288,3:DRAWR 178,0:DRAWR 0,-1
76:DRAWR-178,0:DRAWR 0,176
2090 FOR B=288 TO 112 STEP -35:PLOT 207,B:DRAWR 17
7,0:NEXT B
2100 FOR B=207 TO 384 STEP 36:PLOT B,288:DRAWR 0,-
175:NEXT B
2110 PLOT 193,304:DRAWR 208,0:DRAWR 0,-210:DRAWR -
208,0:DRAWR 0,210
2120 RETURN
```

14. Wordsquare

As has been mentioned earlier, the size of the database is of fundamental importance to the effective execution of many of the programs in this book. The Wordsquare program provides you with an educational way of extending the database by requiring you to make words out of a group of nine letters and adding those words which the Amstrad does not know.

Instructions

When the program is executed the screen shows the initial display followed by the request for the database to be entered. This takes the usual amount of time, and when the internal buffering is complete the nine letters are selected at random. To give the user a reasonable chance of finding at least one word, the computer chooses two vowels and then seven other letters (some of which may be vowels), thus providing a wide choice of letters.

The computer checks its memory to discover how many words it can make, and the number of words is displayed on the screen. After this it is your turn and the computer prompts you to enter a word. Remember that each letter can only be used once.

When the word is completed, pressing the ENTER key will produce the message "ACCEPT (Y/N)?" and if you have made an error or spelt the word incorrectly you should press the N key. Otherwise the Y key is pressed and your total is increased by one. If you can't think of any more words then pressing the green DEL key will pass you on to the next section of the program.

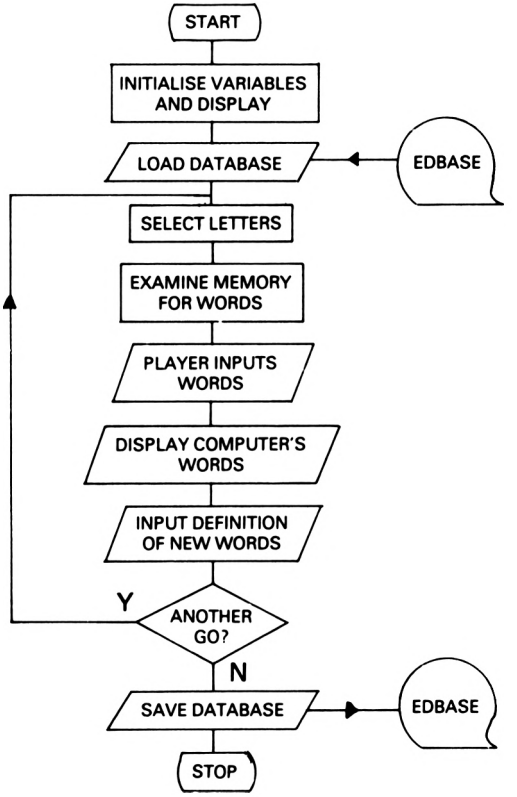
The end routine commences with the computer showing you the words that it managed to find, with the display showing the next word each time you press a key. After this you are given the chance to extend the database by adding the new words which you have formed during the first part of the program. If you cannot provide a definition then the word is forgotten, otherwise you input the definition and this information is remembered for future puzzles.

If new words have been added then the database will need to be resaved, so when you require no more puzzles, a different tape should be placed in the Datacorder to accept the amended database. It is important to keep progressive copies of the database in case any problems occur with a tape or during transfer. This procedure is therefore recommended whenever a new copy of the database has been formed.

The program

Flowchart

The flowchart shows 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$()	The letters
U()	Flag indicating letters used
WD$()	The player's words
CWD$()	The computer's words
NW$()	The new words
WORD$()	Words from the database
CLUE$()	Clues from the database
CS$()	The categories
CT	The number of words the computer has found
PT	The number of words the player has created
RS	Flag set to one if a new word is added to EDBASE
L	The letter under consideration (1-9)
PE	Pen colour
W$	The player's word
NW	The number of words from the database
NC	The number of categories from the database
X,Y	Co-ordinates of the position of a letter in the grid.

Comments on the program

The program is structured, with many of the important steps being written in subroutines. Each of these routines is clearly identified in the listing, and a breakdown of the operation of each section of program is given below.

Lines 10-50
Dimension the arrays and call the routines to construct the initial display.

Line 60
Calls the subroutine to load the database.

Lines 70-120
The letters are selected and the computer's memory examined to

find the words which it can create.

Lines 130-440
The section of code which is used to input the player's word.

Lines 450-770
The routine to display the computer's words is called and then any new words are added before you are asked if you require another puzzle.

Lines 780-990
The routine to create the windows and produce the display on the screen.

Lines 1000-1170
Here the nine letters are chosen at random with at least two vowels.

Lines 1180-1250
This locates the screen position of the character which is to be displayed in the grid.

Lines 1260-1320
The letter is displayed.

Lines 1330-1380
A section of code to update the computer's total when it finds another word.

Lines 1390-1440
A similar section of code to the above which refers to the player's total.

Lines 1450-1650
This is the routine which loads the database. Although the categories and meanings are not required by the program they must be loaded so that they can be repositioned in their correct places when the database is resaved.

Lines 1660-1810
This is the code relating to the search by the computer for some words to be formed from the nine letters.

Lines 1820-1920
This is where the computer's words are shown on the screen.

Lines 1930-2090
If new words have been added to the database then it must be resaved on to a cassette.

The listing

The listing should be entered very carefully and then saved onto a 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 DIM B$(9),U(9),WD$(100),CWD$(100),NW$(100)
20 FOR I=1 TO 9:LET B$(I)=" ":NEXT I
30 CT=0:PT=0:RS=0
40 GOSUB 780
50 GOSUB 1000
60 GOSUB 1450
70 FOR L=1 TO 9
80 LET PE=1
90 GOSUB 1180:GOSUB 1260
100 NEXT L
110 GOSUB 1660
120 GOSUB 1330
130 REM ** INPUT WORD **
140 CLS #3
150 FOR I=1 TO 9:LET U(I)=0:NEXT I
160 LET W$=""
170 FOR L=1 TO 9:GOSUB 1180:PE=1:GOSUB 1260:NEXT L
180 PEN #3,2:PRINT #3:PRINT #3,"    INPUT WORD"
190 LOCATE #3,5,3:PEN #3,1
200 LET M$=INKEY$:IF M$="" THEN GOTO 200
210 IF ASC(M$)=127 THEN GOTO 450
220 IF ASC(M$)=13 THEN GOTO 320
230 FOR L=1 TO 9
240 IF M$=B$(L) AND U(L)=0 THEN GOTO 270
250 NEXT L
260 GOTO 200
270 GOSUB 1180:PE=2:GOSUB 1260
280 U(L)=1
290 PRINT #3,M$;
300 LET W$=W$+M$
310 GOTO 200
320 LOCATE #3,3,4:PEN #3,5:PRINT #3,"ACCEPT  (Y/N)
```

```
?"
330 LET AC$=INKEY$:IF AC$="" THEN GOTO 330
340 IF AC$="N" THEN GOTO 130
350 IF AC$<>"Y" THEN GOTO 330
360 FOR I=1 TO PT
370 IF WD$(PT)=W$ THEN PEN #3,2:PRINT #3,"  ALREAD
Y  GIVEN";:FOR Z=1 TO 2000:NEXT Z:GOTO 130
380 NEXT I
390 PEN #3,2:PRINT #3,"  WORD  ACCEPTED";
400 FOR I=1 TO 1000:NEXT I
410 LET PT=PT+1:GOSUB 1390
420 FOR I=1 TO 1000:NEXT I
430 LET WD$(PT)=W$
440 GOTO 130
450 REM ** FINISH **
460 IF CT<>0 THEN GOSUB 1820
470 LET KK=0
480 FOR J=1 TO PT
490 FOR I=1 TO CT
500 IF WD$(J)=CWD$(I) THEN GOTO 530
510 NEXT I
520 LET KK=KK+1:LET NW$(KK)=WD$(J)
530 NEXT J
540 IF KK<>0 THEN GOTO 630
550 CLS #3:PEN #3,1:LOCATE #3,4,3:PRINT #3,"NO NEW
 WORDS"
560 PEN #3,5:LOCATE #3,1,4:PRINT #3," PLAY AGAIN (
Y/N)"
570 LET AG$=INKEY$:IF AG$="" THEN  GOTO 570
580 IF AG$="N" THEN GOTO 1930
590 IF AG$<>"Y" THEN GOTO 570
600 FOR I=1 TO 9:LET B$(I)=" ":NEXT I
610 CLS #3:GOSUB 1000
620 LET CT=0:LET PT=0:GOSUB 1330:GOSUB 1390:GOTO 7
0
630 REM ** ADD NEW WORDS **
640 FOR I=1 TO KK
650 CLS #3
660 PEN #3,5:PRINT #3:PRINT #3,"  ADD NEW WORDS"
670 PEN #3,2:LOCATE #3,2,3:PRINT #3,NW$(I)
680 PEN #3,5:PRINT #3," DO YOU KNOW  ITS  MEANING
 (Y/N)?"
690 LET W$=INKEY$:IF W$="" THEN GOTO 690
700 IF W$<>"Y" THEN GOTO 760
710 LOCATE #3,2,3:PEN #3,5:PRINT #3," INPUT MEANIN
G...                 "
720 PEN #3,1:LOCATE #3,2,5:LINE INPUT #3,CLUE$(NW+
1)
730 LET WORD$(NW+1)=NW$(I)
740 LET NW=NW+1
```

```
750 LET RS=1
760 NEXT I
770 CLS #3:GOTO 560
780 REM ** INITIAL SCREEN **
790 MODE 0
800 INK 0,20:INK 1,2:INK 2,6:INK 3,18:INK 5,0:INK
6,26:INK 7,24:INK 8,26,6
810 PAPER 7
820 CLS
830 BORDER 24
840 WINDOW #2,4,16,1,3:PAPER #2,3:PEN #2,2:CLS #2
850 PRINT #2:PRINT #2," WORD SQUARE"
860 PLOT 94,399,5:DRAW 514,399:DRAW 514,352:DRAW 9
4,352:DRAW 94,399
870 WINDOW #1,2,8,5,15:PAPER #1,0:CLS #1
880 PLOT 30,336,5:DRAW 256,336:DRAW 256,158:DRAW 3
0,158:DRAW 30,336
890 PLOT 105,336:DRAW 105,158:PLOT 180,336:DRAW 18
0,158
900 PLOT 30,218:DRAW 256,218:PLOT 30,276:DRAW 256,
276
910 WINDOW #3,2,19,19,24:PAPER #3,6:PEN #3,1:CLS #
3
920 PLOT 30,112,5:DRAW 608,112:DRAW 608,14:DRAW 30
,14:DRAW 30,112
930 WINDOW #4,10,20,5,17:PAPER #4,3:CLS #4
940 PLOT 286,336,5:DRAW 638,336:DRAW 638,126:DRAW
286,126:DRAW 286,336
950 PEN #4,1
960 PRINT #4:PRINT #4,"  AMSTRAD":PRINT #4," HAS F
OUND":PRINT #4:PRINT #4,"   0 WORDS"
970 PEN #4,2
980 LOCATE #4,2,9:PRINT #4,"YOU  HAVE":PRINT #4,"
  FOUND":PRINT #4:PRINT #4,"   0 WORDS"
990 RETURN
1000 REM ** CHOOSE LETTERS **
1010 FOR I=1 TO 2
1020 LET P=1+INT(RND(1)*9)
1030 IF B$(P)<>" " THEN GOTO 1020
1040 LET R=1+INT(RND(1)*5)
1050 ON R GOSUB 1130,1140,1150,1160,1170
1060 NEXT I
1070 FOR I=1 TO 7
1080 LET P=1+INT(RND(1)*9)
1090 IF B$(P)<>" " THEN GOTO 1080
1100 LET B$(P)=CHR$(65+INT(RND(1)*26))
1110 NEXT I
1120 RETURN
1130 LET B$(P)="A":RETURN
1140 LET B$(P)="E":RETURN
```

```
1150 LET B$(P)="I":RETURN
1160 LET B$(P)="O":RETURN
1170 LET B$(P)="U":RETURN
1180 REM ** LOCATE PRINTING POSITION **
1190 LET Y=252
1200 IF L<4 THEN LET Y=310
1210 IF L>6 THEN LET Y=194
1220 LET X=54
1230 IF INT((L+1)/3)=(L+1)/3 THEN LET X=130
1240 IF INT(L/3)=L/3 THEN LET X=204
1250 RETURN
1260 REM ** DISPLAY LETTER **
1270 PLOT 640,400,PE
1280 MOVE X,Y
1290 TAG
1300 PRINT B$(L);
1310 TAGOFF
1320 RETURN
1330 REM ** UPDATE COMPUTER´S TOTAL"
1340 PEN #4,1
1350 LOCATE #4,2,5:PRINT #4,USING "###";CT
1360 IF CT=1 THEN LOCATE #4,10,5:PRINT #4," ";
1370 IF CT<>1 THEN LOCATE #4,10,5:PRINT #4,"S";
1380 RETURN
1390 REM ** UPDATE PLAYER´S TOTAL **
1400 PEN #4,2
1410 LOCATE #4,2,12:PRINT #4,USING "###";PT
1420 IF PT=1 THEN LOCATE #4,10,12:PRINT #4," ";
1430 IF PT<>1 THEN LOCATE #4,10,12:PRINT #4,"S";
1440 RETURN
1450 REM ** LOAD DATABASE **
1460 DIM WORD$(1500),CLUE$(1500),CS$(100,2)
1470 CLS #3
1480 PRINT #3:PRINT #3,"  PLACE  CASSETTE   INTO D
ATACORDER     TO LOAD THE        DATABASE"
1490 OPENIN "!EDBASE"
1500 LET I=1
1510 INPUT #9,WORD$(I)
1520 IF WORD$(I)="XXX" THEN GOTO 1550
1530 INPUT #9,CLUE$(I)
1540 LET I=I+1:GOTO 1510
1550 LET NW=I-1
1560 LET I=1
1570 IF EOF THEN GOTO 1620
1580 INPUT #9,CS$(I,1)
1590 INPUT #9,CS$(I,2)
1600 LET I=I+1
1610 GOTO 1570
1620 CLOSEIN
1630 LET NC=I
```

```
1640 CLS #3
1650 RETURN
1660 REM ** FIND WORDS **
1670 CLS #3:PEN #3,8
1680 LOCATE #3,2,3:PRINT #3,"AMSTRAD THINKING"
1690 FOR I=1 TO NW
1700 FOR II=1 TO 9:LET UL(II)=0:NEXT II
1710 FOR J=1 TO LEN(WORD$(I))
1720 FOR K=1 TO 9
1730 IF B$(K)=MID$(WORD$(I),J,1) AND UL(K)=0 THEN
LET UL(K)=1:GOTO 1760
1740 NEXT K
1750 GOTO 1790
1760 NEXT J
1770 LET CT=CT+1:LET CWD$(CT)=WORD$(I)
1780 GOSUB 1330
1790 NEXT I
1800 CLS #3
1810 RETURN
1820 REM ** DISPLAY COMPUTER´S WORDS **
1830 IF CT=1 THEN PEN #3,1:LOCATE #3,2,2:PRINT #3,
"COMPUTER´S  WORD":GOTO 1850
1840 PEN #3,1:LOCATE #3,2,2:PRINT #3,"COMPUTER´S W
ORDS"
1850 PEN #3,5
1860 FOR I=1 TO CT
1870 LOCATE #3,5,4:PRINT #3,"          "
1880 LOCATE #3,5,4:PRINT #3,CWD$(I)
1890 IF INKEY$="" THEN GOTO 1890
1900 NEXT I
1910 CLS #3
1920 RETURN
1930 REM ** RESAVE DATABASE **
1940 IF RS=0 THEN GOTO 1940
1950 CLS #3:PEN #3,5
1960 PRINT #3:PRINT #3,"  PLACE  CASSETTE   INTO D
ATACORDER     TO SAVE THE        DATABASE"
1970 OPENOUT "!EDBASE"
1980 FOR I=1 TO NW
1990 PRINT #9,WORD$(I)
2000 PRINT #9,CLUE$(I)
2010 NEXT I
2020 PRINT #9,"XXX"
2030 FOR I=1 TO NC
2040 PRINT #9,CS$(I,1)
2050 PRINT #9,CS$(I,2)
2060 NEXT I
2070 CLOSEOUT
2080 MODE 1
2090 END
```

15. Pick a Pair

Most children and adults will at some time have played the game of pairs or Pelmanism in which a number of cards are placed face down on the table and the players take it in turn to pick two cards in an attempt to find two the same. This version of the game has an extra twist, in that as well as testing the players' memories it also requires an understanding of words and their meanings.

Instructions

When the program is executed you will be required to enter the number of players in the range 1-4, and then insert the datatape into the recorder. When EDBASE has been loaded and the internal buffering is complete the game will commence with the construction of the board.

The screen is divided into two sections: the large portion to the left displaying the forty-two cards laying face-down and the window to the right showing the number of players and their respective scores.

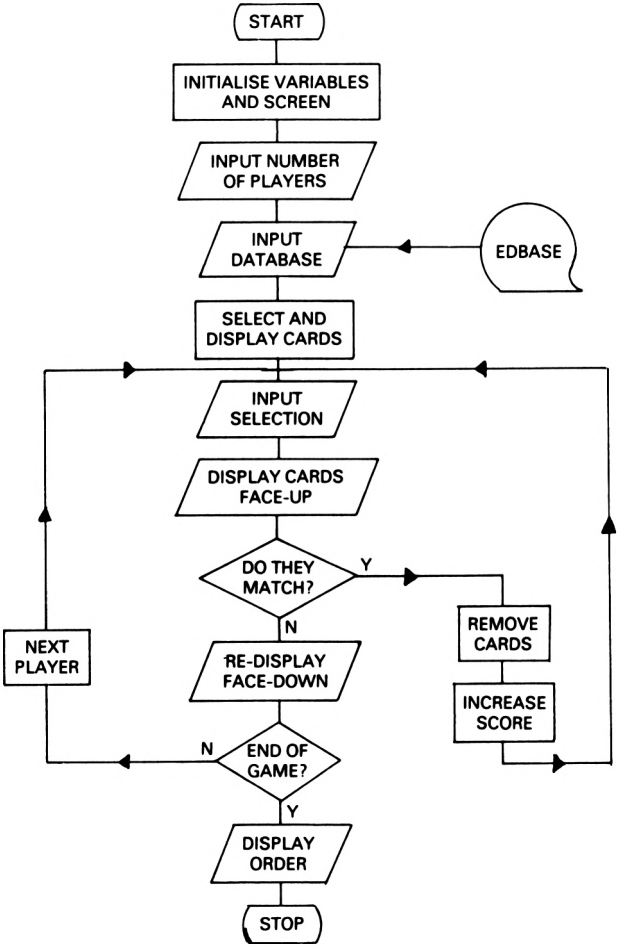
Play commences with player 1 selecting, in response to a message, two cards which he hopes will contain a word and its definition. These cards are then flipped over for all the players to see and if a pair has been correctly selected they are removed from the board, that player's score is increased by ten points, and he is awarded another turn. If, however, the two cards do not contain a correct pair then, after a short delay, they are turned back face-down and play continues with the next player.

This process continues until all forty-two cards have been removed at which time the postions are calculated and displayed on the screen in order.

The program

Flowchart

The flowchart represents the general operation of the program.



Variables

The following table represents the major variables used throughout the program.

A$()	Words in database
B$()	Clues for words
C$()	Words on cards
W$()	List of 21 words
CL$()	List of 21 clues
NW	No. of words in database
Q	No. of players
P	No. of players with current turn
X,Y	Co-ordinate of word in question
PX,PY	Screen co-ordinate of card
PS()	Scores

Comments on the program

The program is very highly structured, with a large number of subroutines. These are clearly labelled throughout the program and described briefly below.

Lines 10-50
Arrays used throughout the program are dimensioned.

Lines 60-270
The section of code which initialises the inks and windows used for the display.

Lines 280-580
This is the main program which controls the game, calling the other subroutines as and when they are required.

Lines 590-610
A short routine to draw a single card on the screen at a position defined by the co-ordinates (X,Y). By changing these values and constantly repeating the routine, all forty-two cards can be constructed.

Lines 620-710
A routine called when a player has made his selection. The cards are turned over and the reverse side is displayed on the screen.

Lines 720-760
A short routine to convert the card number input by a player into a pair of co-ordinates (X,Y) in order that the appropriate card can be manipulated.

Lines 770-800
A simple routine to convert the general co-ordinates (X,Y) into the plot co-ordinates (PX,PY).

Lines 810-840
The co-ordinates (X,Y) are converted for the LOCATE statements.

Lines 850-1040
This routine is called very early in the program, to input the information from EDBASE into the computer. The database is read in a selective manner, ignoring words which have a classification or no definition.

Lines 1050-1230
This is where the words and clues are placed on the cards.

Lines 1240-1300
The routine used to reprint the card face-down when the cards selected do not constitute a pair.

Lines 1310-1330
The scores are updated and displayed on the screen.

Lines 1340-1390
A routine used when a pair has been correctly selected, in order to remove the cards from the board.

Lines 1400-1580
The final section of program, used to discover whether the game is over. If so, the positions are calculated and then displayed 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 DIM A$(1001),B$(10Q1),C$(6,7),W$(21),CL$(21)

20 GOSUB 850
30 BORDER 13:INK 0,13:INK 1,26:PAPER 0:PEN 1
40 FOR X=1 TO 6:FOR Y=1 TO 7:C$(X,Y)="":NEXT Y,X
50 PS(1)=0:PS(2)=0:PS(3)=0:PS(4)=0
60 MODE 2
70 FOR Y=314 TO 0  STEP -48
80 FOR X=0 TO 500 STEP 96
90 GOSUB 590
100 NEXT X
110 NEXT Y
120 LOCATE 10,1:PRINT"     P    I    C   K       A
      P   A   I   R"
130 WINDOW#2,72,80,4,23:PAPER#2,1:PEN#2,0
140 WINDOW#3,1,80,1,2
150 CLS#2
160 LOCATE#2,2,4:PRINT#2,"PLAYER1":PRINT#2:PRINT#2
,"   00"
170 IF Q>1 THEN LOCATE#2,2,8:PRINT#2,"PLAYER2":PRI
NT#2:PRINT#2,"   00"
180 IF Q>2 THEN LOCATE#2,2,12:PRINT#2,"PLAYER3":PR
INT#2:PRINT#2,"   00"
190 IF Q>3 THEN LOCATE#2,2,16:PRINT#2,"PLAYER4":PR
INT#2:PRINT#2,"   00"
200 I=1
210 FOR Y=4 TO 24 STEP 3
220 P=P-1
230 FOR X=2 TO 72 STEP 12
240 LOCATE X,Y:PRINT I
250 I=I+1
260 NEXT X,Y
270 GOSUB 1050
280 FOR P= 1 TO Q
290 CLS#3
300 LOCATE#3,1,1:PRINT#3,"WHICH CARD PLAYER";P;:IN
PUT#3,S$
310 S=VAL(S$)
```

```
320 IF S>42 OR S<1 OR S<>INT(S) THEN GOTO 290
330 I=S:GOSUB 720
340 IF C$(X,Y)="" THEN GOTO 290
350 GOSUB 620
360 I2=I
370 Y1=Y:X1=X
380 CLS#3
390 LOCATE#3,1,1:PRINT#3,"WHICH CARD PLAYER";P;:IN
PUT#3,S$
400 S=VAL(S$)
410 IF S>42 OR S<1 OR S<>INT(S) THEN GOTO 380
420 I=S:GOSUB 720
425 IF C$(X,Y)="" THEN GOTO 380
430 X2=X:Y2=Y
440 GOSUB 620
450 FOR D=1 TO 21
460 IF  W$(D)=C$(X1,Y1) AND CL$(D)=C$(X2,Y2) THEN
GOTO 510
470 IF W$(D)=C$(X2,Y2) AND CL$(D)=C$(X1,Y1) THEN G
OTO 510
480 NEXT D
490 GOSUB 1240
500 GOTO 560
510 GOSUB 1340
520 C$(X1,Y1)="":C$(X2,Y2)=""
530 PS(P)=PS(P)+10
540 GOSUB 1310
550 P=P-1
560 GOSUB 1400
570 NEXT P
580 GOTO 280
590 REM **DRAW CARD**
600 PLOT X+2,Y,1:DRAWR 79,0:PLOT X+83,Y+2:DRAWR 0,
39:PLOT X+84,Y+2:DRAWR 0,39:PLOT X+81,Y+42:DRAWR -
79,0:PLOT X,Y+40:DRAWR 0,-38:PLOT X+1,Y+40:DRAWR 0
,-38
610 RETURN
620 REM ** FILL CARDS **
630 GOSUB 770
640 FOR PY=PY TO PY+40: PLOT PX+2,PY,1:DRAWR 80,0:
NEXT PY
650 GOSUB 810
660 PEN 0:PAPER 1
670 IF LEN(C$(X,Y))<10 THEN GOTO 690
680 LOCATE PX+1,PY:FOR V=1 TO LEN (C$(X,Y)):IF MID
$(C$(X,Y),V,1)=" " AND V-1<10 AND LEN(C$(X,Y))-V<1
0 THEN PRINT LEFT$(C$(X,Y),V-1):LOCATE PX+1,PY+1:P
RINT RIGHT$(C$(X,Y),LEN(C$(X,Y))-V) ELSE NEXT V
690 IF LEN(C$(X,Y))<10 THEN LOCATE PX+1,PY:PRINT C
$(X,Y)
```

```
700 PEN 1:PAPER 0
710 RETURN
720 REM **CONVERT NO. TO X,Y**
730 Y=INT((I-0.1)/6)+1
740 X=((I/6)-(INT(I/6)))*6
750 IF X=0 THEN X=6
760 RETURN
770 REM **CONVERT X,Y TO COORDS**
780 PY=314-(Y-1)*48
790 PX=(X-1)*96
800 RETURN
810 REM **CONVERT TO LOC. X,Y**
820 PX=1+((X-1)*12)
830 PY=1+(Y*3)
840 RETURN
850 REM **START ROUTINE**
860 MODE 0
870 INK 0,2:INK 1,16
880 PAPER 0:PEN 1:BORDER 2
890 INK 2,9,18:PEN 2:LOCATE 1,4:PRINT"    PICK A P
AIR"
900 PEN 1
910 LOCATE 1,12:PRINT" NUMBER OF PLAYERS":PRINT"
  (1,2,3 OR 4) ":PRINT:LINE INPUT"          ";Q$
920 Q=VAL(Q$)
930 IF Q<1 OR Q>4 OR Q<>INT(Q) THEN GOTO 910
940 PRINT:PRINT"INSERT DATA CASSETTE"
950 OPENIN"!EDBASE"
960 I=1
970 INPUT#9,A$(I)
980 IF A$(I)="XXX" THEN GOTO 1020
990 INPUT#9,B$(I):IF B$(I)="*" THEN GOTO 970
1000 I=I+1
1010 GOTO 970
1020 CLOSEIN
1030 LET NW=I-1
1040 RETURN
1050 REM **SET-UP C$()**
1060 FOR X=1 TO 21
1070 LET B=INT(RND*NW)+1
1080 IF LEN(B$(B))>2 AND LEN(A$(B))<10 AND LEN(B$(
B))<10 THEN GOTO 1120
1090 IF LEN(B$(B))<3 OR LEN (B$(B))>18 OR LEN(A$(B
))>18 THEN GOTO 1070
1100 FOR V=1 TO LEN (A$(B)):IF MID$(A$(B),V,1)=" "
 AND V-1<10 AND LEN(A$(B))-V<10 THEN GOTO 1110 ELS
E NEXT V:IF LEN(A$(B))<10 THEN GOTO 1110 ELSE GOTO
 1070
1110 FOR V=1 TO LEN (B$(B)):IF MID$(B$(B),V,1)=" "
 AND V-1<10 AND LEN(B$(B))-V<10 THEN GOTO 1120 ELS
```

```
E NEXT V:IF LEN(B$(B))<10 THEN GOTO 1120 ELSE GOTO
 1070
1120 FOR Y=1 TO X-1
1130 IF W$(Y)=A$(B) THEN GOTO 1070
1140 NEXT Y
1150 W$(X)=A$(B):CL$(X)=B$(B)
1160 NEXT X
1170 FOR I=1 TO 21
1180 X=INT(RND*6)+1:Y=INT(RND*7)+1
1190 IF C$(X,Y)="" THEN C$(X,Y)=W$(I):NEXT I ELSE
GOTO 1180
1200 FOR I=1 TO 21
1210 X=INT(RND*6)+1:Y=INT(RND*7)+1
1220 IF C$(X,Y)="" THEN C$(X,Y)=CL$(I):NEXT I ELSE
 GOTO 1210
1230 RETURN
1240 REM **CLEAR CARD**
1250 X=X1:Y=Y1:GOSUB 770
1260 FOR PY=PY+2 TO PY+40: PLOT PX+2,PY,0:DRAWR 80
,0:NEXT PY
1270 X=X2:Y=Y2:GOSUB 770
1280 FOR PY=PY+2 TO PY+40: PLOT PX+2,PY,0:DRAWR 80
,0:NEXT PY
1290 I=I2:GOSUB 720:LOCATE 2+((X-1)*12),4+((Y-1)*3
):PRINT I2:I=S:GOSUB 720:LOCATE 2+((X-1)*12),4+((Y
-1)*3):PRINT S
1300 RETURN
1310 REM **PRINT SCORES**
1320 LOCATE#2,3,2+(P*4):PRINT#2,PS(P)
1330 RETURN
1340 REM **ERASE CARD**
1350 X=X1:Y=Y1:GOSUB 770
1360 FOR PY=PY   TO PY+42: PLOT PX,PY,0:DRAWR 84,0
:NEXT PY
1370 X=X2:Y=Y2:GOSUB 770
1380 FOR PY=PY TO PY+42: PLOT PX,PY,0:DRAWR 84,0:N
EXT PY
1390 RETURN
1400 REM **END-GAME CHECK**
1410 FOR X=1 TO 6:FOR Y=1 TO 7
1420 IF C$(X,Y)<>"" THEN RETURN
1430 NEXT Y,X
1440 LOCATE 20,9:PRINT"            GAME OVER":PRIN
T:PRINT"                POSITIONS :-"
1450 FOR I=1 TO 4:P(I)=I:NEXT I
1460 FOR X=1 TO 3
1470 FOR A=1 TO 4
1480 IF PS(P(A)) < PS(P(A+1)) THEN Z=P(A):P(A)=P(A
+1):P(A+1)=Z
1490 NEXT A
```

```
1500 NEXT X
1510 FOR X=1 TO Q
1520 PRINT"                                    ";X;":- PLA
YER";P(X)
1530 NEXT X
1540 PRINT:PRINT"                                    PLA
Y AGAIN ?"
1550 Q$=INKEY$:IF Q$="" THEN GOTO 1550
1560 IF Q$="Y" THEN GOTO 30
1570 IF Q$<>"N" THEN GOTO 1550
1580 END
```

Appendix
EDBASE print-out

You may occasionally want to obtain a print-out of the complete database in order to locate any errors which may have occurred or definitions with which you are no longer satisfied. The program shown below will produce a hard copy of the complete EDBASE database on the Amstrad printer or any similar peripheral.

```
10 DIM A$(1001),B$(1001),C$(50),CL$(50)
20 REM ** PRINT EDBASE **
30 CLS
40 PRINT TAB(5)"EDBASE PRINT-OUT"
50 PRINT TAB(5)"================"
60 PRINT
70 PRINT "INPUT DATA TAPE"
80 PRINT
90 PRINT
100 OPENIN "EDBASE"
110 LET I=1
120 INPUT#9,A$(I)
130 IF A$(I)="XXX" THEN GOTO 170
140 INPUT#9,B$(I)
150 LET I=I+1
160 GOTO 120
170 LET J=1
180 IF EOF THEN GOTO 230
190 INPUT#9,C$(J)
200 INPUT#9,CL$(J)
210 LET J=J+1
220 GOTO 180
230 PRINT:PRINT
240 PRINT "PLEASE WAIT. BUFFERING IN PROCESS"
250 CLOSEIN
260 PRINT#8,TAB(13)"WORDS WITH CLASSIFICATIONS"
270 PRINT#8,TAB(13)"========================="
280 PRINT#8
290 WIDTH 60
300 ZONE 15
310 FOR K=1 TO I-1
320 IF LEN(B$(K))<>2 THEN GOTO 340
```

```
330 PRINT#8,A$(K),B$(K),
340 NEXT K
350 PRINT#8:PRINT#8
360 PRINT#8,TAB(20)"CLASSIFICATIONS"
370 PRINT#8,TAB(20)"==============="
380 PRINT#8
390 FOR K=1 TO J-1
400 PRINT#8," ",C$(K),CL$(K)
410 NEXT K
420 PRINT#8:PRINT#8
430 PRINT#8,TAB(20)"WORDS WITH CLUES"
440 PRINT#8,TAB(20)"================"
450 PRINT#8
460 FOR K=1 TO I-1
470 IF LEN(B$(K))<3 THEN GOTO 490
480 PRINT#8," ",A$(K),B$(K)
490 NEXT K
500 PRINT#8:PRINT#8
510 PRINT#8,TAB(20)"SINGLE WORDS"
520 PRINT#8,TAB(20)"============"
530 FOR K=1 TO I-1
540 IF B$(K)="*" THEN PRINT#8,A$(K),
550 NEXT K
```

When the program is executed you will be required to input the database cassette and after the usual period of buffering the file will be printed on the screen in the format used in Chapter 3.

Duckworth Home Computing

AN EDUCATIONAL DATABASE FOR THE AMSTRAD
by Richard Hurley & David Virgo

A collection of entertaining programs, designed for children of nine upwards. The earlier chapters introduce the reader to the database management system which is used as the source of information for most of the programs. The rest of the book contains the educational programs, including a geographical puzzle which creates a graphical display of the World or Great Britain and requests the user to locate certain countries, capitals or towns, a crossword puzzler which will provide an endless supply of crosswords, and a card game which requires the user to construct words from a hand of cards containing different letters. There are also programs to develop mathematical ability, including a horse-racing game and a version of snakes and ladders.

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.

ISBN 0-7156-1977-2

9 780715 619773

Duckworth
The Old Piano Factory
43 Gloucester Crescent, London NW1

ISBN 0 7156 1977 2

IN UK ONLY £6.95 NET

AN EDUCATIONAL DATABASE FOR THE AMSTRAD

RICHARD HURLEY & DAVID VIRGO

AMSTRAD CPC

MÉMOIRE ÉCRITE
MEMORY ENGRAVED
MEMORIA ESCRITA

https://acpc.me/