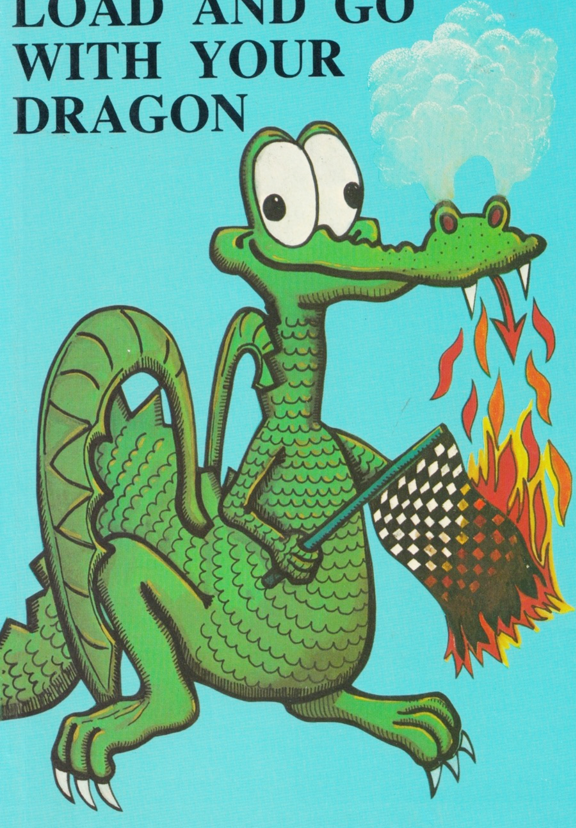
LOAD AND GO WITH YOUR DRAGON



LOAD AND GO WITH YOUR DRAGON

BY

JOHN PHIPPS

AND

TREVOR TOMS

PHIPPS ASSOCIATES

99 East Street
Epsom
Surrey KT17 1EA

ISBN Ø 95Ø 73Ø2 9 7

This book was created using WordStar on a Panasonic JD85Ø with Diablo 63Ø printer and Courier 1Ø typeface.

First Edition January 1983

Printed and bound by:
Redwood Burn Ltd.,
Yeoman Way,
Trowbridge,
Wiltshire

CONTENTS

Our thanks are due to our program testers for service sometimes beyond the call of duty. Any remaining glitches are, of course, our own.

Cover Design by S. Lewis

INTRODUCTION

This book is intended to be fun but also educational. It contains programs for the Dragon 32 home micro-computer, which is equipped with Microsoft 16K Colour BASIC. The majority of the programs are short (about 5Ø lines) so that it is easy to type them in and "RUN". Some are serious and some are silly but we hope most of all that you will find them fun. There are also some longer ones towards the back of the book, which might be more difficult to get going but should prove more rewarding. The complexity of a program increases with the square of its length and most commercial programs are over 2ØØØ lines long!

The book is educational in several ways. Firstly, we provide home-work tools like "Sums Tester" and "Vocabulary Tester", which might make humdrum tasks more exciting. Secondly, a major section of the book is devoted to explaining in a comprehensive way the Graphics features on the Dragon. This could prove a useful supplementary source to your BASIC manual for understanding these very powerful features. Lastly, comprehensive chapters are included on "Debugging Programs" and "Hints and Tips" both of which should help to get your own programs running as well as the programs in this book.

It would be wrong to allow anyone to think that typing in programs and running them is "programming". It is not. However, the study of worked examples is an essential learning aid for anyone beginning to program. So often, a manual will teach the words and the grammar without giving any real understanding of how to connect them together to make a story or a poem. All the examples in this book are comprehensively annotated with comments and explanations and every encouragement is given for the reader to modify them to see how minor changes would affect their running. A good understanding of the principles of program construction will help immeasurably when you start to construct your own programs, which we hope that you will do.

The title "Load and Go" is a term of art referring to compilers or assemblers which not only process the program but also start to execute it immediately the compilation or assembly is finished. In terms of the Dragon, perhaps it should be "Type and Run", but "Load and Go" has a certain ring to it.

Not all the material in this book is completely new. We have gathered together the old favourites (golden oldies) from our other works (see back page). In the process, most have had to be totally re-written to take advantage of the powerful features of the Microsoft BASIC language - nearly always the resulting program is much shorter than the original, which is a striking testimony to the power of the Dragon interpreter. About half of the material is totally new and written specially for the Dragon.

It was very tempting to try to include some "Gee Whiz" super programs which emulate arcade games. Such programs are great fun but usually have to be written in machine code to get them to work fast enough. Once this is done, their educational value tends to be small because they cannot be studied easily.

Where possible, we have tried to introduce games that can be played by two people as well as the computer and we start and end the book with such a program. When you want to show friends how the computer works, it is more fun if they can also take part, instead of having to watch in awed amazement as you expertly play the keyboard.

We have also deliberately used only features to be found on the standard Dragon. To ensure this, we purchased the Dragon used in the preparation of the book over the counter in a chain store. Although the programs do use colour and having a colour television will add an extra dimension for you, we have not assumed its use. The presentation of all of the programs will be quite intelligible in black and white. We have not assumed the use of joysticks, which at the time of writing were in short supply, nor of printers, nor of extra memory beyond the standard, nor disk units.

We suggest, however, that at an early stage you start using a cassette recorder to retain the programs on tape as you enter them. A special section on how to get this working is included in "Hints and Tips". A cassette recorder is far from being an optional extra as some publicity implies - it is quite essential.

How this book was produced

How this book was produced

First we produce a working version of each program and test it thoroughly with junior volunteers who attempt to test it to destruction. To ensure that this code is transferred exactly to print we have built a parallel to serial converter which is attached to the printer port of the Dragon and used to transfer the program listings into our word processor.

Now suppose, as sometimes happens, that you put in the program and type "RUN" and it doesn't work. Firstly, "Don't Panic". Secondly, list it on your printer or on the screen and check against the book very carefully; also ask someone else to check it with you. If this doesn't show the problem, then read the chapter on debugging, which might help to pin down the area in which the problem is occurring. If all else fails, please write to us and describe the problem. Sending a cassette of your version of the program is helpful as is sending a printer listing of the program. Those who send a self-addressed stamped envelope also get higher priority than those who don't!

Program presentation

The programs have been annotated in two stages. The first stage was when they were originally prepared on the Dragon. A first stage annotation would look like this:

```
1Ø REM NIM PROGRAM
```

This type of remark, which has a line number attached, you should always put into your version. Beware of leaving it out because that line number may be used as the destination of a GOTO or a GOSUB statement.

The second stage annotation was added in the word processor and might look like this:

```
1Ø   FOR Z=1 TO 1Ø:A(Z)=Ø:NEXT           'reset the array
```

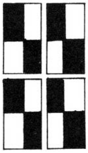
The part following the ' character is an explanatory remark added later. This type of remark can be omitted when putting in the program. Include it if you wish, but we have been very liberal with this type of remark so as to make the programs as clear as possible.

PAIRS

A game for two players, where the object is to collect as many pairs of cards as possible. The whole pack is spread out and each player may turn over two cards at a time. If they match, then that player keeps the pair, removes it from the field of play and is allowed another go. If the cards do not match, then they are turned face down again, and it is the other player's turn. If you have a good memory, then you can benefit from your opponent's mistakes.

To turn over a card you enter the reference code shown against the card on the screen e.g. "B2" and that card will be displayed. It shows the card suit (C,D,H,S) followed by the denomination (A,K,Q,J,T,9 ...). A pair clearly is made up of two the same, e.g. two Aces, two Kings. A problem peculiar to computers arises if the player attempts to turn over a card that is no longer there or attempts to match a pair by turning over the same card twice. In such cases, the computer responds by printing 'HO HO" in the space!

A card is represented by the chequer pattern CHR$(134) in which the colour is orange and the background black. It should look like this:



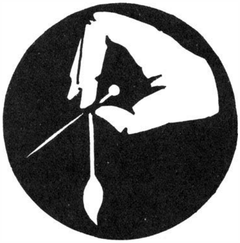
```
10 CLS1:DIM P(52)      'Array P contains the pack
20 PRINT TAB(10);"PAIRS"       'print the rules
30 PRINT:PRINT"THE OBJECT OF THE GAME IS TO "
40 PRINT"GET AS MANY PAIRS AS POSSIBLE."
50 PRINT"EACH PLAYER GETS TWO TURNS."
60 PRINT"IF THE TWO CHOICES MATCH THEN"
70 PRINT"YOU GET ANOTHER GO."
80 PRINT"OTHERWISE YOUR OPPONENT GETS "
90 PRINT "A GO."
                       'get the names of the players
100 INPUT "FIRST PLAYER'S NAME";A$(0)
110 INPUT"SECOND PLAYERS'S NAME";A$(1)
                       'shuffle and set up the pack
120 PRINT"PLEASE WAIT A MOMENT -"
130 PRINT"I AM SHUFFLING THE PACK."
140 FOR X=1 TO 52
150 Z=RND(52)
160 IF P(Z)<>0 THEN 150 ELSE P(Z)=X
170 NEXT X

180 REM DRAW OUT THE PACK      'five rows 4 of 11 and 1 of 8
190 CLS1:CL=112        'alter setting of CL for different colour
200 FOR X=0 TO 4
210 FOR Y=0 TO 10
220 Z=X*11+Y+1:IF Z=53 THEN 310
230 IF P(Z)=0 THEN 290    'blank card
240 P=X*96+Y*3            'calculate card position
250 X$=CHR$(134+CL)       'set up card image
260 PRINT@P,X$+X$;:PRINT@P+32,X$+X$;     'print card
270 PRINT@P+64,CHR$(Y+65);     'row letter
280 PRINTUSING"#";X+1;         'grid reference
290 NEXT Y
300 NEXT X
```

```
310 S(0)=0:S(1)=0          'set scores to 0
320 PS=0
330 PRINT@480 ,S(W);"/";S(1-W);A$(W);"'S GO       ";
340 GOSUB 470:GOSUB 560        'get a card ref and turn it over
350 PS=PP                      'PS=first card turned
360 GOSUB 470:GOSUB 560        'get a second card and turn over
370 GOSUB 720                  'test for a pair
380 IF PR THEN P(PP)=0:P(PS)=0      'set these cards as gone
390 IF PR THEN PRINT@411,"YES";ELSE PRINT@411,"NO";
400 PRINT@409,"";:INPUT G$     'wait for Enter key to be pressed
410 GOSUB 650                  'turn card back again
420 PP=PS:GOSUB 650            'turn second card back
430 IF PR THEN S(W)=S(W)+1     'add to score
440 IF NOT PR THEN W=1-W       'change over to other player
450 IF S(0)+S(1)>=26 THEN 790 'if all 26 prs gone then end
460 GOTO 320
        'get a card reference
470 PRINT @409,"";:INPUT G$
480 IF LEN(G$)<2 THEN 550
490 G1=ASC(G$)-65              'G1 becomes 0-10 for row
500 IF G1<0 OR G1>10 THEN 550
510 G2=ASC(MID$(G$,2))-49      'G2 becomes 0-4 for ref
520 IF G2<0 OR G2>4 THEN 550
530 PP=G2*11+G1+1:IF PP>52 THEN 550      'PP=pack reference
540 RETURN
550 SOUND 80,5:PRINT @409,STRING$(6,32);:GOTO 470
        'turn over the card referenced by PP
560 REM TURN OVER PP(1-52) G1(0-10) G2(0-4)
570 P=G2*96+G1*3
        'ST is the suit CT=denomination
580 Z=P(PP)-1:ST=INT(Z/13):CT=Z-ST*13+1:ST=ST+1
590 IF P(PP)=0 OR PP=PS THEN 630   'blank card or two the same
600 PRINT@P,MID$("CDHS",ST,1)+CHR$(143);
610 PRINT@P+32,CHR$(143)+MID$("AKQJT98765432",CT,1);
620 RETURN
     'error ?
630 PRINT@P,"HO";:PRINT@P+32,"HO";
640 RETURN
```

```
650 REM TURN CARD BACK PP=CARD
660 IF P(PP)=Ø THEN X$=CHR$(143) ELSE X$=CHR$(134+CL)
670 PP=PP-1:P1=INT(PP/11)           'work out column and row
680 P2=PP-P1*11:P=P1*96+P2*3
690 PRINT@P,X$+X$;
700 PRINT@P+32,X$+X$;
710 RETURN

720 REM TEST FOR PR
730 IF P(PS)=Ø OR P(PP)=Ø THEN PR=Ø:RETURN
740 IF P(PS)=P(PP) THEN PR=Ø:RETURN
750 A=P(PS):A=A-INT(A/13)*13        'denomination for PS
760 B=P(PP):B=B-INT(B/13)*13        'denomination for PP
770 IF A=B THEN PR=-1 ELSE PR=Ø
780 RETURN
        'end of game
790 PRINT @48Ø,"GAME OVER"
800 PRINT"FINAL SCORE"
810 PRINT A$(Ø);S(Ø)
820 PRINT A$(1);S(1)
830 INPUT"ANOTHER GAME";A$
840 S(Ø)=Ø:S(1)=Ø         'reset the scores
850 IF LEFT$(A$,1)="Y" THEN 12Ø
```

PIN THE TAIL ON THE DONKEY

Here's a piece of fun to test your timing - try to pin the tail on the donkey. You press a key, and the length of time that the key is held down determines the length of the tail. The trick is to hold it down for exactly the right length of time, or to do it with the fewest number of attempts. When the tail reaches the donkey the score is given. If you wait too long, though, you can cause the donkey some damage.



```
10 L=65280           'peek location
20 CLS1:PRINT "PIN THE TAIL ON THE DONKEY";
30 PN=RND(10)+180         'random position for donkey
40 PRINT@448,"PRESS A KEY"
50 PRINT @PN,CHR$(34);    'ears
60 PRINT @PN+30,CHR$(128)+CHR$(128)+CHR$(128)+"O" 'body,head
70 PRINT@PN+62,CHR$(129);CHR$(131);CHR$(130);      'body,legs
80 PRINT' @PN+94,CHR$(135);CHR$(143);+CHR$(139);    'legs
90 P=192    'START OF TAIL
100 G=0
110 GOSUB 170        'get the time interval
120 T=INT(T/2)
130 PRINT @P,STRING$(T,131);        'tail of length T
140 P=P+T:G=G+1:IF P>PN+30 THEN PRINT@448,
         "OUCH THAT HURT !!!!":GOTO 220
150 IF P=PN+30 THEN PRINT @448,
         "WELL DONE - YOU TOOK ";G;"GOS":GOTO 220
160 GOTO 110
```

```
          'key pressing routine
170 IF PEEK(L)<>255 THEN 170  'wait for enter key to go up
180 IF PEEK(L)=255 THEN 180   'wait for new key depression
190 TIMER=Ø                   'set timer
200 IF PEEK(L)<>255 THEN 200  'wait for key to go up again
210 T=TIMER:RETURN            'set T

220 INPUT"ANOTHER GO";A$
230 IF LEFT$(A$,1)="Y" THEN 1Ø
```

The routine above is for detecting a key depression. We might have used INKEY$ function for this but it would only give one keystroke for each key depression, and would not reflect the duration of the stroke. For example try this program:

```
1Ø A$=INKEY$
2Ø IF A$="" THEN 1Ø
3Ø PRINT "KEY DOWN"
4Ø GOTO 1Ø
```

This will print one line for every time that you press a key, but it will make no difference whether you hold that key down or just peck at it.

We use this technique again in the next program "Ski Run" where a fast reaction is required to keys being pressed. Who needs joystick buttons?

SKI RUN

If you always wanted to try the slalom runs in the Winter Olympics, here's your big chance. You must guide yourself down the slope avoiding the posts on the way. Use the -> key for right and the @ key for left. Hold your finger on the key for rapid continuous movement. The program contains two courses, which are chosen at random. After the program listing you will find instructions on how you can alter the courses or enter some new ones of your own.

```
10 REM SKI RUN
20 GOTO 90

30 REM CHECK FOR MOVEMENT
40 D=PEEK(65280):IF D=255 THEN RETURN   'no key being pressed
50 IF D=&HDF THEN S=S+1  '->KEY
60 IF D=&HFB THEN S=S-1  '@KEY
70 S=ABS(S):IF S>31 THEN S=31 'ensure skier stays on the screen
80 RETURN

90 REM INITIALISE
100 GOSUB 350                'print rules
110 GOSUB 300                'select the course
120 P=VAL(MID$(C$,1,2))      'course start position
130 S=VAL(MID$(C$,3,2))      'skier start position
140 H=0                      'H=total posts hit
```

```
15Ø REM RUN THE COURSE
16Ø PRINT
17Ø FOR X=5 TO LEN(C$)-1 STEP 2          'C$ contains the course
               'note the ";" characters are important
               'P=left hand edge of course G= width of course
               'S=skier starting position
18Ø PRINT @48Ø+P,"*";:PRINT @48Ø+P+G,"*";:PRINT @48Ø+S,"V";
19Ø PRINT @511,""                'cause a screen scroll
2ØØ GOSUB 3Ø                     'check for key pressed
21Ø P=P+VAL(MID$(C$,X,2))        'take next two characters of course
                                 'evaluate and add to value of P
22Ø GOSUB 3Ø                     'check for movement again
23Ø IF S<=P OR S>=P+G THEN H=H+1    'if outside course limits then
                                    'add one to hits total
24Ø FOR D=1 TO DL:NEXT D            'control speed
25Ø NEXT X
               'drop through when string all done i.e.
               'course over
26Ø REM  FINISH LINE
27Ø PRINT @48Ø+P,STRING$(G,"-")
28Ø PRINT "YOU HIT ";H;"POSTS"
29Ø END

3ØØ REM SELECT COURSE
31Ø C$="1417ØØØØØØØØ-1-1-1-1-1ØØØØØØØ1Ø1Ø2ØØ-1-1-1ØØØ1Ø2ØØ
     -1-1-1-1-1ØØØØ-2ØØ-1-1-1ØØØØØØØØØØ1Ø1Ø1Ø1Ø1-2-2-2
     ØØØØØØØØØ1Ø1Ø1Ø1Ø1Ø1Ø1Ø1Ø2Ø2Ø2ØØØØØØØØØØØØØ-1-1-1Ø1
     Ø1Ø1-1-1-1-1-1-2-2-2-2-2ØØØØØØØØØØØØ1Ø1Ø1Ø1Ø1Ø1-2-2
     -2-2ØØØØØØØ1Ø1Ø2Ø2Ø2Ø2Ø1Ø1Ø1ØØØØØØØØØØØØØØØØØØØ"
32Ø X=RND(2): IF X=1 THEN RETURN
33Ø C$="ØØØ2ØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØ1Ø1Ø1Ø1Ø1Ø1Ø1Ø1Ø1
     Ø1ØØØØØØØØØØ2Ø2Ø2Ø2ØØØØ-1-1-1-1-1-1-1Ø1Ø1Ø1Ø1Ø1ØØ
     ØØØØØØØØØØØØØØØØØØ-1-1-1-1Ø2Ø2Ø2Ø2Ø1Ø1Ø1-2-2-2-2-2
     -2-2-1-1-1-1-1Ø1Ø1-1-1ØØØØØØØØØØØØ"
34Ø RETURN
```

```
350 REM RULES
360 CLS1:PRINT TAB(8);"***SKI RUN***"
370 PRINT
380 PRINT"YOU MUST STAY IN BETWEEN THE "
390 PRINT"COURSE POSTS ALL THE WAY DOWN"
400 PRINT "IF YOU HIT THE POSTS OR MOVE"
410 PRINT"OUTSIDE THE COURSE, YOU WILL"
420 PRINT"LOSE POINTS. INITIALLY YOU MAY"
430 PRINT"SELECT THE LEVEL OF DIFFICULTY"
440 PRINT"YOU WISH TO ATTEMPT."
450 PRINT "USE -> KEY FOR RIGHT"
460 PRINT "AND @ FOR LEFT"
470 PRINT "WHAT LEVEL ARE YOU?"
480 PRINT "1=RANK AMATEUR"
490 PRINT "2=AVERAGE SKIER"
500 PRINT "3=DASHING EXPERT"
510 INPUT SKILL
520 IF SKILL>Ø AND SKILL<4 THEN 54Ø
530 CLS1:PRINT"SORRY? -":GOTO 47Ø
540 G=3+(4-SKILL)*2'WIDTH OF RUN
550 DL=(4-SKILL)*5Ø'CONTROL SPEED
560 PRINT"ARE YOU READY? -";
570 FOR X=3 TO 1 STEP -1            '3.2.1 start
580 PRINT X;"..";
590 FOR D=1 TO 5ØØ:NEXT D
600 NEXT X
610 RETURN
```

Course selection is performed in the routine starting at line 3ØØ. You may prefer to ask for a course number to be entered as the program is started, or just to replace the two courses given here. After the routine has executed, C$ must contain a course description in the following format:

Characters

1-2 Starting position of left hand post. The right hand post position is calculated from the degree of skill entered. The width of course is G which can take on a maximum value of 9 for the amateur. So, course start position should not be greater than 21.

3-4 Starting position of skier - this would normally be the post starting position plus two. To be very annoying you could start the skier outside the posts altogether. Maximum value is width of screen (31).

5-6 The rest of the string is a series of two digit numeric values which represent the values to be added to the current post position. Do not use values greater than Ø2 or -2 as the player cannot move the skier more than two positions in each turn.

A small but valid course description would be:

```
C$="15 17 ØØ ØØ Ø1 Ø1 -1 -1 -2 -2 ØØ ØØ"
```

The spaces are just put in to make it easier to read. It says the starting position of the left post is 15, the skier position is 17. For the first two turns the course runs straight, then it bends to the right for two turns and to the left for two turns, then more sharply to the left. The last two turns are straight.

The program will automatically show a finish line and the number of hits made. Since the speed of the game is approximately three turns per second, an ideal course should take about 3Ø seconds to run fully. The maximum length course that you can construct is dictated by the maximum string length of 255. This means that the maximum number of turns is 254/2 (because the number must be even) less 2 i.e. 125 turns.

Some people have difficulty entering the very long strings for the course correctly. If you miss out one number in say the string "ØØ Ø2 Ø1 Ø1" so that it becomes "ØØ 2Ø 1Ø 1" it is taken as +2Ø then +1Ø with an odd one left at the end. The odd one left causes an error report, but meanwhile the course goes out of control. To check that you have typed it correctly, add the following lines:

```
7ØØ GOSUB 3ØØ
71Ø FOR Z=5 TO LEN(C$)-1 STEP 2
72Ø PRINT VAL(MID$(C$,Z,2))
73Ø NEXT Z
```

Then execute GOTO 7ØØ in direct mode and you will see the course values printed out for you in pairs. If any of the values is greater than plus or minus 2, then you have missed something out. Remember however that there are two courses. Run this two or three times until you are sure that both are correct.

SUMS GALORE

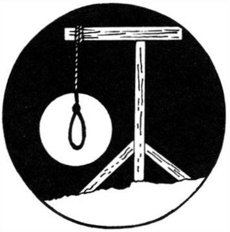
It may not be current educational theory, but I doubt whether any parent would disagree that regular practice, table learning, and table tests are the foundations of good arithmetic. It can be very difficult for the teacher to make up enough sums to keep the children active and busy, so here is an automatic sums tester. An example program of this type is given in the Dragon manual, but this has significant advantages. It is shorter to type in, it does not assume knowledge of directed (negative) numbers, which may not have been covered at primary school level, and it dodges the question of remainders after division, by making sure that the division sums always give a whole number quotient with no remainder.

A maximum number can be specified at the beginning. So, for example, to test number bonds up to a sum of 1Ø only, specify 5 as the maximum number. This will generate sums up to 5+5 maximum. To test number bonds involving carry, specify 9, this gives a maximum sum of 9+9. To test all tables up to 12 times 12, specify 12 as the maximum and so on.

There are always ten questions, and the answers wrong or right stay on the screen at the end, so the teacher can look at them to see what mistakes are being made. Appropriate reinforcement is also given according to the score. We suggest that the teacher should set up the problem specifying the maximum and the rule to be used, and then come back when the test is complete.

```
10 REM SUMS TESTER
20 CLS1:PRINT TAB(8);"SUMS TESTER":PRINT
30 INPUT"WHAT IS THE LARGEST NUMBER";MX      'see text
40 INPUT"WHICH RULE DO YOU WISH(+-*/)";A$    'see text
50 R=INSTR("+-*/",A$):IF R=0 THEN 40         'invalid reply
         'T=total questions C=correct answers  W=wrong
60 T=0:C=0:CLS1:W=0:N=1:M=MX
70 M=M/10:IF INT(M/10)<>0 THEN N=N+1:GOTO 70   'how many digits?
80 N$=STRING$(N,"#")+" "       'edit mask for printusing commands
90 T=T+1:IF T>10 THEN 310      'always 10 questions only
100 N1=RND(MX):N2=RND(MX)      'pick the numbers to be used
110 IF R=1 OR R=3 THEN 170     'plus or multiply
120 IF N2>=N1 THEN 100         'ensure that result positive
130 IF R=2 THEN 170
140 N2=RND(INT(MX/3)):IF N2=0 THEN 100  'fix division so that
                                        'result is integer
150 N1=INT(N1/N2)*N2:IF N1=0 THEN 100   'avoid asking about 0
160 IF N1=N2 THEN 100          'not n/n which is too easy
170 ON R GOTO 180,190,200,210       'plus,sub,mult,divide
180 Q$="PLUS ":DS=N1+N2:GOTO 220    'DS is the correct answer
190 Q$="MINUS ":DS=N1-N2:GOTO 220
200 Q$="TIMES ":DS=N1*N2:GOTO 220
210 Q$="DIVIDED BY":DS=INT(N1/N2)
220 PRINT"WHAT IS ";
230 PRINTUSING N$;N1;
240 PRINT Q$;:PRINT USING N$;N2;    'ask the question
250 INPUT AN                        'get the answer
260 P=(T-1+W)*32+30:IF P>480 THEN P=480  'which line are we on?
270 PRINT @P,"";
280 IF AN=DS THEN PRINT"OK"; ELSE PRINT "NO THE ANSWER WAS";DS
290 IF AN=DS THEN C=C+1 ELSE W=W+1
300 GOTO 90
         'print the total at the end
310 PRINT:PRINT"YOU SCORED";C;"OUT OF";T-1
320 IF C=T-1 THEN PRINT"GO TO THE TOP OF THE CLASS"
330 IF C>T-3 THEN PRINT"WELL DONE"
340 IF C<INT(T/5)THEN PRINT"OH DEAR - WE NEED MORE PRACTICE"
```

HANGMAN



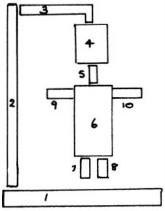
This ever popular game is played with two people (or one with an extremely short memory) - one to enter the selection of words and another to guess the words under computer control. The words are entered as a DATA statement in line 120, and there can be as many as you wish. Guessing takes place by entering either a letter or a whole word. A table at the top of the screen is updated as letters are used up and any correct are displayed in the word at the bottom of the screen. If you guess at the whole word, then the letter table is not updated - your guess is either right, in which case, you win, or wrong, in which case you lose a life. You have 10 lives. As these are lost a picture of a hanging man is gradually built up on the screen.

```
100 DIM A$(26)
110 REM CHANGE WORDS BELOW FOR NEW GAME
               'first data element is the number of words
120 DATA 5,"WORD","HORSE","DOG","CAT","GUESS"
130 READ W  'CONTROL VARIABLE
140 CLS0:READ W$:W=W-1:E=0           'W is the word count
                                     'E=lives lost
150 FOR Z=1 TO 26:A$(Z)=CHR$(64+Z):NEXT Z   'set up letter table
160 PRINT @405,STRING$(LEN(W$),"*");    'word mask at bottom
170 FOR Z=1 TO 26:PRINT @1+Z,A$(Z);:NEXT Z   'letter table
180 PRINT @448,"GUESS LETTER OR WORD    ";STRING$(4,8);:INPUT A$
190 IF A$="" THEN 180          'invalid
200 IF LEN(A$)>1 THEN 320    'you have guessed at the whole word
                               'set this letter as used
210 A=ASC(A$)-64:IF A>0 AND A<26 THEN A$(A)=" " 'note space!
```

```
220 T$=""
230 FOR A=1 TO LEN(W$)
240 L=ASC(MID$(W$,A))-64
250 IF A$(L)=" " THEN L$=MID$(W$,A,1) ELSE L$="*"
260 PRINT @404+A,L$;      'print word mask
270 T$=T$+L$              'T$=current word mask
280 NEXT A
290 IF T$=W$ THEN 340    'all letters now correct
300 IF INSTR(W$,A$)=0 THEN 360     'letter not in word
310 GOTO 170
320 REM GUESS AT WHOLE WORD
330 IF A$<>W$ THEN 360   'w$ is the correct answer
340 PRINT @448,"YOU WON !! YOU LOST ";E;"LIVES"
350 GOTO 410
360 REM LOST A LIFE
370 E=E+1:CL=112          'E=lives lost CL=colour
380 IF E<10 THEN ON E GOSUB 450,460,470,480,490,500,
                            510,520,530:GOTO 170
390 GOSUB 540       'last life lost
400 PRINT@448,"OH DEAR - THE WORD WAS ";W$
410 INPUT"ANOTHER GAME";A$
420 IF LEFT$(A$,1)<>"Y" THEN END
430 IF W=0 THEN RESTORE:GOTO 130   'start word list again
                                   'if it is exhausted
440 GOTO 140

                   'different entry for each life lost
        'print base
450 FOR X=363 TO 371:PRINT@X,CHR$(143+CL);:NEXT X:RETURN
        'print pole
460 FOR X=332 TO 140 STEP-32:PRINT @X,CHR$(133+CL);:NEXT X:RETURN
        'print cross piece
470 FOR X=141TO 143:PRINT@X,CHR$(140+CL);:NEXT X:
             PRINT @144,CHR$(141+CL);:RETURN
        'head
480 FOR X=176 TO 177:PRINT@X,CHR$(143+CL);:NEXT X:RETURN
        'neck
490 PRINT @208,CHR$(133+CL);:PRINT@209,CHR$(138+CL);:RETURN
```

```
          'body
500 Y$=CHR$(143+CL):PRINT@240,Y$;:PRINT @241,Y$;
               :PRINT@272,Y$;:PRINT@273,Y$;:RETURN
          'left leg
510 PRINT @304,CHR$(138+CL);:RETURN
          'right leg
520 PRINT @305,CHR$(133+CL);:RETURN
          'left arm
530 PRINT @239,CHR$(140+CL);:RETURN
          'right arm
540 PRINT @242,CHR$(140+CL);:RETURN
```

This is a picture of how the complete figure should look:



SILLY QUIZ

Ever fancied yourself as a budding Nicholas Parsons? This quiz program looks after all the chores, like keeping score and deciding whether the answer is on the card. This has always been a popular favourite, because you can invent your own questions and make up your own version.

We include it because it is a striking example of the power of the INSTR function in Microsoft BASIC. In other languages this function always has to be simulated; here it is part of the BASIC command set. Suppose that the question was "Who is the British Prime Minister?". Now, the possible answers could be:

Margaret Thatcher
Maggie Thatcher
Mrs. M. Thatcher
Mrs. Margaret Thatcher
Mrs. Thatcher
Thatcher
Thatcher, I think??
The Iron Lady
Wedgewood Benn   (of all these - this is the only wrong one!)

You can see from this that people not used to computers can put in all sorts of extraneous data like commas, full stops, "I think", and so on. It makes the computer look pretty stupid if the correct answer is, say "Surrey" and if it marks "County of Surrey", which is of course equally correct, as wrong.

We resolve the problem by searching the answer for the key word, or words that tell you the answer is correct. For the question above, the key word that must be there is "Thatcher". Depending on your politics, you might also allow "Margaret" or "Iron Lady", so these are alternative answers.

The INSTR function works by searching string A to see if it contains within its length string B. So, the statement:

```
W=INSTR(A$,B$)
```

would return results as follows, depending on the state of A$ and B$.

A$	B$	W
"DRAGON"	"RAG"	2
"DRAGON"	"DRAG"	1
"DRAGON"	"CAT"	Ø
"DOWNING"	"OWN"	2
"DOWNING"	"IN"	5
"DOWNING"	"DOG"	Ø

If Ø is returned, this means that A$ does not contain B$. A number returned is the number of the starting position of B$ within A$.

Because the number of answers is variable, we adopt a convention to tell the computer when we have exhausted all the possible answers. The last answer is followed by a ";" character - this tells us that the next data statement is a new question. The word "FINIS" tells us that all the questions have been exhausted.

```
10 REM SILLY QUIZ PROGRAM
11 REM FORMAT IS QUESTION,ANS1,ANSN;
20 DATA "WHAT IS THE CAPITAL OF CANADA","OTTAWA;"
21 DATA "WHAT IS A BABY HARE CALLED","LEVERET;"
22 DATA "WHO IS THE GREATEST","DAD","MUM","I AM;"
23 DATA "HOW MANY DAYS IN JULY","31;"
24 DATA "WHAT IS SHAKESPEARE'S FIRST NAME","WILLIAM",
         "BILL","FRANCIS;"
25 DATA "WHERE IS THE DRAGON COMPUTER MADE",
         "SWANSEA","WALES","UK;"
26 DATA "WHERE DOES THE BRITISH PRIME MINISTER LIVE","DOWNING;"
27 DATA "WHO IS THE PRESIDENT OF THE USA","REAGAN;"
199 DATA "FINIS"
               'here starts the program proper
200 S=Ø:Q=Ø     'S=number correct Q=number of questions
210 READ A$:IF A$="FINIS"THEN 350  'test for end of questions
220 FOR X=1 TO 1Ø:Z$(X)="":NEXT 'clear out the answer slots
230 PRINT A$:LINEINPUT RP$     'print question get answer
240 X=1:OK=Ø:Q=Q+1:DN=Ø  'DN means all answers done
                         'OK set means correct answer found
                         'X= number of possible answers
250 READ AN$
        'if the answer is the last one set DN and
        'strip off the ";" character
260 IF RIGHT$(AN$,1)=";" THEN DN=-1:AN$=MID$(AN$,1,LEN(AN$)-1)
270 IF INSTR(RP$,AN$)<>Ø THEN OK=-1      'INSTR test for answer
280 Z$(X)=AN$:X=X+1
290 IF NOT DN THEN 250
300 IF OK THEN PRINT"THAT IS CORRECT":S=S+1:GOTO 21Ø
310 PRINT "THAT IS NOT CORRECT"
320 PRINT"IT SHOULD BE ";:FOR Z=1 TO X-1:PRINT Z$(Z);",";:NEXT
330 PRINT
340 GOTO 21Ø

350 REM END OF QUIZ
360 PRINT "YOU SCORED ";S;"OUT OF ";Q
370 IF S<Q/2THEN PRINT "OH DEAR!"
380 IF S>=Q-1 THEN PRINT"WELL DONE"
390 IF S<Q/1ØTHEN PRINT"STAND IN THE CORNER!"
```

DEBUGGING PROGRAMS

So you've just finished writing your new masterpiece and you're holding your breath while you type RUN...

Then...nothing. The program just sits with a blank screen, or an error report is given at the foot of the screen. What now? Where do you start to unravel the reason why the Dragon has decided not to play ball?

First and foremost, write down the error message on a scrap of paper. The reason for this is that it is too easy to forget the line number that has been printed as part of the message.

If the program just "hangs up", press BREAK to determine whereabouts the program has gone wrong. You will get a message "BREAK IN 6Ø", for example, meaning that it was on line 6Ø at the time.

Also, do not type CLEAR - this removes all the variables in the program, and these can be extremely important in helping you to discover errors. Be very careful also not to use the Editor to change any of the lines until you have discovered what you need to know. Using the Editor automatically resets all the variables and any further investigation after that will be fruitless. Using LIST is alright and causes no damage.

If you're a newcomer to programming it can sometimes be difficult to understand just how a program can go wrong. What I am going to do is to give you a small program which has some errors in it. You will enter the program, run it, then along with this text, we shall (hopefully) find out where the mistakes occur, correct them, then try again. In this way, you should pick up the main points in trying to debug your own programs.

We want to write a program that drops a stone from the top of the screen to the bottom. When it hits the bottom, the Dragon is to print "SPLAT". We want the stone to drop in different places each time, so we need a random starting point along the top line.

For the stone we use the letter "O". So here is the first attempt at the program:

```
10 RND(31)
20 FOR Y=0 TO 32
30 PRINT @Y*31+XR;"O"
40 NEXT Y
50 PRINT "SPLAT"
```

First of all we get ?SN ERROR IN 10. This means Syntax error, so we have broken the rules of grammar. Looking at the card, we can see that RND is a function not a command - this means that it can only appear after an "=" command. So the first line should be:

```
10 X=RND(31)
```

RUN again.

Now we get ?SN ERROR 30

Referring to the reference card again, you can see that the PRINT @ statement uses a "," character not a   So use Edit to correct this mistake, and line 30 becomes:

```
30 PRINT @Y*31+XR,"O"
```

Run again and we get a stream of "O"s running down the screen, diagonally, and the result ?FC ERROR IN 30

FC means an illegal function call, so we have done something wrong. Use PRINT to find out the values of Y and XR in direct mode, that is without a line number e.g.:

```
PRINT Y          'you type this
 17              'dragon replies
OK
PRINT XR         'you type this
 Ø               'dragon replies
OK

PRINT Y*31+XR    'you type this
 527             'dragon replies
OK
```

This looks odd for a start; there are 32 lines on a screen aren't there? No, there aren't. So our loop should only run up to 16. Secondly XR should be a random number in the range 1-31, this is what we asked for in line 1Ø. "Ø" can't be right. We set the variable as X not XR; we have used the wrong name. Clearly also, we have run off the screen and this is why we got the FC error - a rethink is called for:

```
1Ø X=RND(31)
2Ø FOR Y=Ø TO 16
3Ø PRINT @Y*31+X,"O"
4Ø NEXT Y
5Ø PRINT "SPLAT"
```

?FC ERROR IN 3Ø again!

```
PRINT Y,X,Y*31+X        'you type
     16             2Ø
     516                          'dragon replies
OK
```

Yes there are sixteen lines. Why are we still off the screen? Ah! Inspiration - Ø to 16 is actually 17 isn't it? Line 2Ø becomes:

```
2Ø FOR Y=Ø TO 15
```

At last we get OK, no syntax errors. But, the "O" still runs diagonally across the screen. The whole thing would be clearer without the clutter on the screen, so let's put in a clear screen statement as line Ø5:

```
Ø5 CLS
```

It is still going too fast for us to see what is going on so let's put a PRINT and STOP statement in as line 32:

```
32 PRINT @36Ø,X;Y;Y*31+X;:STOP
```

We get the first "O" printed then:

```
21 Ø 21                         (this appears at the bottom so as
BREAK IN 32                     (not to confuse the display at the
OK                              (top - this is why we used
                                (PRINT @36Ø,
```

That looks alright, so we type CONT (for continue) and we get:

Another "O" printed to the left of the first and:

```
21  1  52                       (appears over the top of the last
BREAK IN 32                     (try
OK
```

X has stayed the same, Y has increased by one, as we would expect, and the position has increased by 31. We obviously have the wrong multiplier for Y. We have made the same mistake again, but the other way this time. Yes, the last column on the screen is numbered 31 but there are 32 in all, so line 3Ø should read:

```
3Ø PRINT @Y*32+X,"O"
```

Great! It drops straight this time. Now take out line 32 again and re-test; the stone should now drop all the way down the screen without BREAKs. This trick of using STOP and CONT is a very powerful way of helping you to debug a program. First of all, it slows the program down and gives you time to inspect the screen, secondly, you can print any of the variables to check if they have the correct values.

Now we realise that we have made a mistake in the design. As each new position of the stone is printed we should wipe out the old one, to give the impression of a stone falling. So we need to add a line to wipe out the old stone by printing space at the line before. Now, there is no line before line Ø so we must watch out for that.

```
Ø5 CLS
1Ø X=RND(31)
2Ø FOR Y=Ø TO 15
22 IF Y<>Ø THEN PRINT @(Y-1)*32+X,"
3Ø PRINT @Y*32+X,"O"
4Ø NEXT Y
5Ø PRINT "SPLAT"
```

Fine. Now it might be nice if the stone fell a little more slowly so we could see it fall and really the splat should appear where the stone fell and not at the right hand side, so we add:

```
32 FOR DL=1 TO 5Ø:NEXT DL
```

and

```
5Ø PRINT TAB(X);"SPLAT"
```

If you run the program now, it should do exactly what we required. You should have seen how we tracked down the errors, using three important techniques:

1. Knowing roughly what the program is to do, otherwise you are fighting "blind".

2. Using direct PRINT commands (i.e. without a line number) to check the value of variables when an error occurs.

3. Using STOP commands to allow portions of the program to be checked in more detail. These can be removed once you are

happy that the section is working properly.

There are a couple of final points I would like to make here - write your programs down on paper before you enter them. In this way, it is easier to spot typing mistakes when you run the program, since you can check incorrect lines (ones that give error reports) against your own copy. Secondly, if you have a printer - USE IT! Take a fresh listing of your program. You wouldn't believe how many errors are caused by programmers looking at an out-of-date listing.

Just before we finish, I'll give a list of the program as it should look after all the changes above have been included:-

```
Ø5 CLS
1Ø X=RND(31)
2Ø FOR Y=Ø TO 15
22 IF Y=Ø THEN 4Ø ELSE PRINT @(Y-1)*32+X," "
3Ø PRINT @Y*32+X,"O"
32 FOR DL=1 TO 5Ø:NEXT DL
4Ø NEXT Y
5Ø PRINT TAB(X);"SPLAT"
```

The line numbers are a bit untidy, so this is a good point to renumber the program before saving it onto tape. Here's a good Mnemonic for the RENUM command - RENUMber NOW.

N	new number that you want the line to become
O	old number where you want to start
W	width of the interval between numbers

So the RENUM command becomes:

```
RENUM 1Ø,5,1Ø          (call the new first line 1Ø
                       (start with the old line Ø5
                       (leave a gap of 1Ø between numbers
                       (so that they become 1Ø,2Ø,3Ø ....
```

PUTTING

You have managed to reach the green, now it's time to putt the ball. The hole is represented by the letter"O" and the ball position is shown by "X". Naturally the playing surface is coloured green. You are asked to enter the direction of the shot as a number 1-12 as on the hands of a clock and the strength to be applied. The new position of the ball is then calculated by trigonometry and the ball wobbles its way to the new spot. If the new position of the ball and the hole coincide, then it goes "PLOP". If you hit the ball into the rough at the edge of the screen, then you lose a penalty stroke and the ball is dropped for you at the edge of the green.

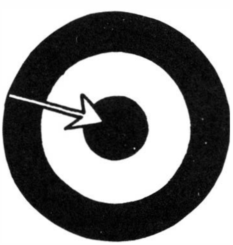
```
1Ø CLS1
2Ø INPUT"DO YOU WANT THE RULES";A$
3Ø IF LEFT$(A$,1)="Y" THEN GOSUB 52Ø 'print the rules
4Ø PI=3.141952:HT=Ø      'note no function for pi
5Ø HX=RND(32)-1          'starting position of hole
6Ø HY=RND(14)
7Ø TX=RND(32)-1          'starting position of ball
8Ø TY=RND(14)
9Ø IF TX=HX AND HY=TY THEN 5Ø 'must not win immediately!
1ØØ GOSUB 35Ø            'recalculate the hole position
11Ø GOSUB 36Ø            'recalculate the ball position
12Ø PRINT@HP,"O";
13Ø PRINT @TP,"X";
        'get the direction data
14Ø GOSUB 39Ø:INPUT"DIRECTION";XD:IF XD<1 OR XD>12 THEN 14Ø
        'get the strength data
15Ø GOSUB 39Ø:INPUT"STRENGTH";XS
```

```
170 XD=15-XD:IF XD=12 THEN XD=Ø    'turn direction into right
                                   'size and value to express
                                   'as pi/6 radians
180 NX=TX+INT(COS((PI/6)*XD)*XS+.5)      'work out new x
190 NY=TY-INT(SIN((PI/6)*XD)*XS+.5)      'work out new y
200 RG=Ø                           'RG true means in the rough
210 IF NY<1 THEN NY=1:RG=-1        'line Ø is out of bounds
220 IF NY>15 THEN NY=15:RG=-1
230 IF NX<Ø THEN NX=Ø:RG=-1
240 IF NX>31 THEN NX=31:RG=-1
250 GOSUB 4ØØ:HT=HT+1
260 IF RG THEN PRINT@48Ø,"YOU ARE IN THE ROUGH";:HT=HT+1
270 TX=INT(TX):TY=INT(TY)
280 PRINT@HP,"O";:PRINT@TP,"X";
290 IF HY=TY AND HX=TX THEN PRINT@HP,"PLOP" ELSE 12Ø
        'putt now sunk - print the result
300 PRINT "YOU TOOK";HT;"PUTTS"
310 IF HT<2 THEN PRINT"HOLE IN ONE - DRINKS ON YOU"
320 IF HT>4 THEN PRINT"OH DEAR"
330 IF HT>8 THEN PRINT"WHIST IS A GOOD GAME"
340 END
        'recalculate the absolute hole position
350 HP=32*HY+HX:RETURN
        'recalculate the absolute ball position
360 TP=32*INT(TY)+INT(TX)
370 IF TP=511 THEN TP=51Ø          'to stop the screen roll
380 RETURN
        'clear the top line of the screen
390 PRINT @Ø,"":PRINT @Ø,"";:RETURN
        'move the ball
400 REM MOVE X FROM TX,TY TO NX,NY
410 X1=ABS(NX-TX):X2=ABS(NY-TY)
420 IF X1>X2 THEN X=X1 ELSE X=X2   'X is the number of steps
                                   'required for the move
430 SX=(NX-TX)/X:SY=(NY-TY)/X      'SX SY are the increments
440 PRINT@TP," ";             'blank out the old ball position
450 IF ABS(NY-TY)<.5 THEN SY=Ø     'test if there
460 IF ABS(NX-TX)<.5 THEN SX=Ø
470 IF SY=Ø AND SX=Ø THEN RETURN   'ball move finished
```

```
480 TX=(TX+SX):TY=(TY+SY):GOSUB 360
490 PRINT @TP,"X";              'print new ball position
500 FOR D=1 TO 50:NEXT D        'slow down ball movement
510 GOTO 440

        'print the rules
520 CLS1:PRINT TAB(8);"PUTTING"
530 PRINT:PRINT"THE HOLE IS SHOWN AS ";CHR$(34);"O";CHR$(34)
540 PRINT"YOUR POSITION IS SHOWN AS ";CHR$(34);"X";CHR$(34)
550 PRINT"ENTER DIRECTION AS ON THE HANDS OF A CLOCK 1-12"
560 PRINT"ENTER STRENGTH AS A NUMBER UP TO ABOUT 20"
570 PRINT:PRINT"GOOD LUCK"
580 PRINT:PRINT"PRESS ENTER TO CONTINUE"
590 INPUTA$:CLS1:RETURN
```

TARGET PRACTICE

As you run past a stationary target, you must fire your arrow to try and hit it. The Dragon goes so fast it is not that easy to do. To make it easier we allow a speed handicap - enter a number up to about 1Ø0 to slow it down to human speed, but see how small a number you can use and still hit it. The number of times you run past is counted as well as the number of shots you need to take.

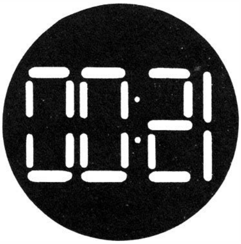


```
1Ø REM TARGET PRACTICE
2Ø INPUT"HANDICAP (Ø-1ØØ)";D2     'D2 slows down the dragon
3Ø CLS1
4Ø N=Ø:M=1                 'M=number of shots N=no of turns
5Ø T=RND(12):X=RND(31)     'T,X define the target position
6Ø PRINT @T*32+X,"O";      'O is the target
7Ø N=N+1
8Ø FOR Y=Ø TO 13           'run past
9Ø IF Y THEN PRINT@(Y-1)*32," ";    'Y not =Ø blank out the
                                    'old position
1ØØ PRINT@Y*32,">";
11Ø IF INKEY$<>"" THEN 16Ø          'test to see if key has
                                    'been pressed
12Ø FOR D=1 TO D2:NEXT D            'delay to slow it down
13Ø NEXT Y
14Ø PRINT@13*32," ";                'blank out the one at
                                    'the bottom
15Ø GOTO 7Ø
```

```
                                         'fire off the arrow
160 FOR Z=Ø TO X:PRINT@Y*32+Z,">";:NEXT Z
170 IF Y=T THEN 21Ø                      'has it hit the target?
180 FOR Z=Ø TO X:PRINT @Y*32+Z," ";:NEXT Z
19Ø M=M+1
2ØØ GOTO 7Ø
                               'Yes it has hit the target
21Ø PRINT @T*32+X,"*";:PRINT @448,"YOU GOT IT!  YOU TOOK ";
          M;"SHOTS IN ";N;"TURNS"
22Ø INPUT "ANOTHER GAME";A$
23Ø IF LEFT$(A$,1)="Y" THEN RUN
```

This game is rather like the example we used for the chapter on Debugging Programs. Compare it to see how the extra facilities have been put in. The statement LEFT$(A$,1) (see line 23Ø) is included so that you can answer either "YES" or "Y" and it will have the same effect. The command "RUN" executed within a program doesn't cause it to try to eat its own tail as you might think - it simply causes all variables to be reset as if you had typed "RUN" at the keyboard. Try adding a handicapping facility so that the final score depends on both the speed, the number of turns and the number of shots (such as 5 times the speed + the number of turns + number of shots) to encourage people to try the higher speeds.

DIGITAL CLOCK



A digital clock that plays Westminster Chimes - now there's a novelty! It uses the timer inside the Dragon to get the ticks and displays the hours, minutes and seconds, using the 24 hour clock notation. If you have set the chiming mechanism, then except between the hours of 1Ø pm and 7 am it will chime the hours for you. Naturally it only chimes on the twelve hour clock because 23 bongs would be rather tiresome.

The main problem with this program is in tuning or regulating it. The timer could vary slightly from machine to machine and in any case doesn't give precise one second ticks. One is added into the timer count about every 22.43 milliseconds, which makes for a difficult conversion into seconds. A basic tick is set at 44 strokes of the timer, but further regulation is required by adjusting the variable TL - this has been set for our machine at 1.Ø13 which keeps good time to within 1 or 2 seconds per hour. Bear in mind that your machine could be slightly different and might require further regulation.

The display of the time should look like this:

```
*************

HH : MM : SS

*************
```

```
10 REM DIGITAL CLOCK
20 TK=44:TL=1.013'REGULATOR
30 M$="## : "               'print mask
40 INPUT"CHIMES ON (Y/N)";C$
50 INPUT"ENTER TIME (HHMM)";T$           'enter as e.g 1400(Enter)
60 IF LEN(T$)<4 THEN 50
70 H=VAL(MID$(T$,1,2))                   'get the hours
80 M=VAL(MID$(T$,3,2))                   'get the minutes
90 CLS1:S=0
100 TIMER=0                              'reset the timer
110 PRINT @329,STRING$(12,"*")           'top line
120 PRINT @393,STRING$(12,"*")           'bottom line
130 IF TIMER<TK THEN 130                 'wait until count ->44
140 S=S+TL:IF S>=59.5 THEN S=S-60:M=M+1 'add in tl worth of secs
150 IF M>=60 THEN M=M-60:H=H+1:GOSUB 220 'adjust minutes/hours
160 IF H>=24 THEN H=0          'special action at midnight
170 PRINT@361,"";              'move the cursor
180 PRINTUSING M$;H;           'print hours
190 PRINTUSING M$;M;           'print minutes
200 PRINTUSING "##";S;         'print seconds
210 GOTO 100
                   'chime subroutine
                   'not within unsocial hours
220 IF H>22 OR H<7 THEN RETURN
                   'test if chimes on or not
230 IF LEFT$(C$,1)<>"Y" THEN RETURN
240 IF H>12 THEN CH=H-12 ELSE CH=H 'only chime up to 12
                   'A$=chime music repeated twice
250 A$="T3L2;O2;ECD;O1;L1;G;L2G;O2DEL1C;"
260 PLAY A$:PLAY A$
270 FOR Z=1 TO CH           'bongs
280 PLAY "P4T3O1GP4"
290 NEXT
300 S=S+17+CH*2.2           'catch up the seconds allowing
                            'for the right number of bongs
310 RETURN
```

The variable S, the seconds counter, never gets reset to Ø at the end of a minute. It merely has 6Ø deducted from it. This is so that the regulator can work. If it were reset to zero, then the fractional part of the tick regulator would always be lopped off.

Notice the use of a PRINT USING mask set at line 3Ø as "## : ". This is a common trick to shorten the PRINT USING statements. It is tedious if you have to say every time:

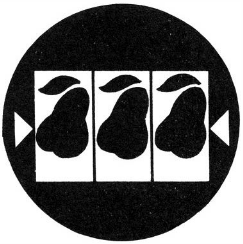
```
PRINTUSING "## : ";H
```

PRINTUSING M$;H is much quicker to type. Print masks can contain ordinary symbols like ":" and " ", which are carried straight through into the edited line. You can use this trick to put spaces between numbers, where they are closely packed.

If the result being printed is not quite a whole number, then PRINT USING will round it off for you. For example, 58.6 seconds would print as 59, when edited in this manner. For this reason, you may occasionally see "-Ø" being printed in the seconds column and this indicates that the rounding is taking place.

It is also worth noting the use of the PLAY command. It catches most people out to find that L2 actually means a half length note and not a double length one as you might expect. This is very clearly set out in the manual but you would not be the only one to be fooled, if you get it wrong.

FRUIT MACHINE



If an ordinary fruit machine is called a "one armed bandit", then this has to be the "no-armed bandit". There are no traditional wheels inside the machine. A combination of characters is generated and displayed at the bottom of the screen. You have a set period of time to press a key and "hold" the machine. Two of a kind pays 5. Three of a kind pays 5Ø. Two stars at the beginning pays 1Ø and one star at the beginning pays 2. Anything else loses 1Ø. See if you can stay ahead of the game. To help a bit, you can set the speed at the beginning. High numbers go slower; a number of about 2Ø gives a reasonable speed. The whole game lasts for 3Ø seconds.

```
1Ø REM FRUIT MACHINE
2Ø CLS1:TIMER=Ø             'reset the time counter
3Ø PRINT"HIGH NUMBERS GO SLOWER"
4Ø S=Ø:INPUT"SPEED";F       'get the input speed
5Ø Z$=""                    'Z$ is the output line
6Ø FOR X=1 TO 3
                            'choose 3 at random from the string
7Ø Z$(X)=MID$("$*+-=.#",RND(7),1)
8Ø NEXT X
```

```
90 FOR X=1 TO 3:PRINT Z$(X);" ";:NEXT X 'print the line
100 FOR D=1 TO F                        'wait to see if inkey
110 RP$=INKEY$:IF RP$<>"" THEN 150      'if key hit work out
                                        'the payoff
120 NEXT D
130 PRINT
140 GOTO 50

150 W=-10                     'default score
160 IF TIMER>=1200 THEN 230   'see if game ended
170 IF Z$(1)="*" THEN W=2     'one * only
                              'test for two of a kind
180 IF Z$(1)=Z$(2) OR Z$(2)=Z$(3) OR Z$(1)=Z$(3) THEN W=5
                              'test for two * at left hand side
190 IF Z$(1)=Z$(2) AND Z$(1)="*" THEN W=10
                              'test for three of a kind
200 IF Z$(1)=Z$(2) AND Z$(2)=Z$(3) THEN W=50
                              'print pay line
210 PRINT "PAYS";W:S=S+W
220 GOTO 50                   'repeat
                              'end of game - print score
230 PRINT"TIME UP  ";:IF S>0 THEN PRINT "YOU GOT";S:END
                              'overall loss position
240 PRINT "YOU LOST";ABS(S)
250 PRINT TAB(6);"I ACCEPT IOUS"
```

NIM

The origins of the game Nim are obscure, but it is believed to have come from the Orient some time in the way distant past. It is played with matchsticks or stones, which are divided into three piles. Each player in turn takes any number but from only one pile at a time. The object of the game is to ensure that you take the last match and not your opponent.

The theory of the game is based on binary numbers. Consider the case of one left in each of two piles - clearly if your opponent takes one then you can take the other and win. So 1,1,Ø is a winning situation. Arguing from this you can see that 5,5,Ø is also a winning situation, because however many he takes, you take the same number; and this follows for any number i.e. n,n,Ø.

When a third pile is introduced you have a problem, but if you strive for a balanced situation by making sure that all the binary bits of the sum add up to 2 or to Ø then you will always win. Take for example a situation of 3,2,1. In binary this is expressed as:

Pile	Number	Binary
1	3	1 1
2	2	1 Ø
3	1	Ø 1
	Sum	2 2

This is a balanced situation, from which you always win. If he takes any number the appropriate reply is shown below:

<u>He takes from pile n</u>	<u>You take from pile n</u>
3 from pile 1	1 from pile 2 giving 1,1,Ø again
2 from pile 1	2 from pile 2 giving 1,1,Ø
1 from pile 1	1 from pile 3 giving 2,2,Ø
2 from pile 2	2 from pile 1 giving 1,1,Ø
1 from pile 2	3 from pile 1 giving 1,1,Ø
1 from pile 3	1 from pile 1 giving 2,2,Ø

So you always win!

When you play against the computer, if you make the right first move you will usually win. There can be up to 29 matches in any pile, which makes the calculations a bit more tricky and one slip and you're done for.

Many years ago, about 195Ø, I built a machine for playing this game out of War Surplus bits and pieces. It was a monstrous creation with meters, bits of old packing cases and decimal to binary decoders made out of strips of tin, with sellotape wrapped over them at intervals. It made the simplistic assumption, that the opening move would always be a pre-emptive strike on one whole pile, to simplify the calculations. By comparison with the Dragon, this creation was beyond the pale, but I have retained a strong affection for the game, which should be better known.

```
1Ø REM NIM PROGRAM
2Ø CLS1:PRINT TAB(12);"NIM"     'rules
3Ø PRINT:PRINT"THE OBJECT OF THE GAME IS TO "
4Ø PRINT"TAKE THE LAST MATCH.  YOU MAY "
5Ø PRINT"TAKE ANY NUMBER, BUT FROM ONLY"
6Ø PRINT"ONE PILE AT A TIME."
7Ø PRINT:INPUT"PRESS ENTER TO CONTINUE";A$
8Ø FOR N=1 TO 3:N(N)=RND(29):NEXT        'set pile values
```

```
          'display board - test for game over
90 GOSUB 140:GOSUB 210:
          IF G THEN PRINT"SOME PEOPLE HAVE IT - - -":END
          'get a move from you - redisplay the board - test end
100 GOSUB 240:GOSUB 140
110 GOSUB 210:IF G THEN PRINT"OH DEAR, YOU SEEM TO HAVE WON":END
120 GOSUB 290:GOTO 90          'work out best move

140 REM SR TO DISPLAY BOARD
150 CLS1:FOR N=1 TO 3
160 PRINT:PRINT N;
170 IF N(N)=Ø THEN 190
                    'chr$(133) is a green and black stripe
180 FOR Z=1 TO N(N):PRINT CHR$(133);:NEXT Z
190 PRINT:NEXT N
200 PRINT:RETURN

210 REM SR TO TEST IF GAME OVER
220 FOR Z=1 TO 3:IF N(Z)>Ø THEN G=Ø:RETURN
230 NEXT Z:G=-1:RETURN

240 REM SR TO GET MOVE FROM YOU
250 INPUT"ROW";R:IF R<1 OR R>3 THEN 250
260 INPUT"HOW MANY";A:IF A<1 OR A>N(R) THEN 250
270 N(R)=N(R)-A
280 RETURN

290 REM SR TO DECIDE MOVE
300 FOR B=1 TO 3               'for each pile in turn
310 IF N(B)=Ø THEN 360
320 FOR M=1 TO N(B)        'test whether pile-M would balance
330 GOSUB 410              'test if bits balance
340 IF C=2 OR C=Ø THEN N(B)=N(B)-M:RETURN     'yes it does
350 NEXT M
360 NEXT B
                    'drop through if no balanced situation can
                    'be found
```

```
370 GOSUB 510          'make a rude remark - take one off any
                       'non-zero pile
380 FOR B=1 TO 3:IF N(B)>0 THEN N(B)=N(B)-1:RETURN
390 NEXT B
400 RETURN

410 REM SR TO TEST FOR EQUALITY
420 T=1:FOR A=1 TO 5             'T is doubled every time
                                 'to test one bit at a time
                                 '5 iterations because 2 to power
                                 '5 is 32 and max size is 29
430 C=0:FOR N=1 TO 3
440 IF B=N THEN T2=N(N)-M ELSE T2=N(N)
450 IF T2 AND T THEN C=C+1       'AND function works as if
                                 'T is an integer variable
460 NEXT N
470 IF C=1 OR C=3 THEN RETURN
480 T=T+T              'double T for next bit
490 NEXT A
500 RETURN             'exit through this route means C=0 or C=2

510 REM RUDE REMARKS
520 RR=RR+1:IF RR>5 THEN RETURN
530 ON RR GOTO 540,550,560,570,580
540 PRINT"I THINK YOU HAVE A CHANCE":GOTO 590
550 PRINT "YOU HAVE PLAYED THIS GAME BEFORE":GOTO 590
560 PRINT"I DON'T BELIEVE THIS":GOTO 590
570 PRINT"OK SMARTY PANTS":GOTO 590
580 PRINT"I THINK I WILL TAKE UP CHESS"
590 FOR DL=1 TO 1500:NEXT DL        'wait so that remark
                                    'can be seen
600 RETURN
```

SPACE DOCKING

"Mothership to Snoopy - start docking procedures - acknowledge - over".
"Snoopy to Mothership - we have you on visual - starting docking procedures - over and out".
You are driving Snoopy and have to dock with the mothership. The mothership is moving forwards in space and from a standing start, you have to match speeds and distance so that you arrive, travelling at the same relative speed and at the same place. Go too fast, and you will zoom past, never to be seen again. Go too slow, and you will exceed the time limit. Use too much fuel, and you will run out and drift for ever.

First select the time for which you wish your rockets to fire. Enter a negative time, e.g. -2, to indicate two seconds of negative thrust, which will decrease your speed. If as a result your speed goes negative you will also go backwards not forwards!

You can, if you want, idle for a specified period. To do this, enter the burn time as zero and then the program will ask for the idle time. If your speed is faster than the mothership, then you will continue catching it up. If your speed is slower than the mothership, then idling will increase the distance between you.

Each second of real time, the state of play is shown on the parameter display screen. This shows:

Time - the elapsed time from start. The maximum permitted is 5Ø seconds.
Speed - the actual speed of Snoopy in space. You start from zero.
Approach speed - the relative speed between you and the mothership. Negative numbers means that you are slipping behind.
Range - the distance between you and the mothership. Negative numbers mean that you have gone too far.
Fuel - the amount of fuel remaining. You have 5Ø units to start with, and each second of burn uses up one unit.

```
1Ø CLS1:PRINT TAB(1Ø);"SPACE DOCKING"
2Ø PRINT:PRINT"YOU HAVE TO DOCK WITH YOUR"
3Ø PRINT"PARTNERS SHIP. YOU HAVE 5Ø "
4Ø PRINT"SECONDS. TELL ME HOW LONG TO"
5Ø PRINT"BURN (USE E.G.-2 FOR NEGATIVE "
6Ø PRINT"THRUST). TO IDLE, GIVE BURN "
7Ø PRINT"DURATION OF Ø, AND I WILL ASK "
8Ø PRINT"FOR THE IDLE PERIOD. NEGATIVE "
9Ø PRINT"RANGE MEANS YOU HAVE GONE PAST"
1ØØ PRINT "YOUR PAL."
         'F=fuel T=elapsed time V=Snoopy speed
         'U=mothership speed Y=distance travelled by snoopy
         'H=distance apart at the beginning
11Ø F=5Ø:T=Ø:V=Ø:Y=Ø:H=1ØØ:U=1Ø
12Ø INPUT"HOW LONG TO BURN";S
13Ø A=SGN(S)
14Ø D=ABS(S)
15Ø IF D<>Ø THEN 18Ø
16Ø INPUT"IDLE FOR HOW LONG";D
17Ø D=ABS(D)
18Ø D=D-1
19Ø IF D<Ø THEN 12Ø
2ØØ T=T+1:V=V+A:F=F-(A*A)
21Ø IF F<Ø THEN 29Ø       'out of fuel
```

```
220 Y=Y+V:H=H+U:CLS1:GOSUB 370          'display parameters
230 IF T>49 THEN 360                    'time up
240 IF ABS(Y-H)>5 THEN 180              'close enough ?
250 IF ABS(V-U)>2 THEN 180              'speeds matched ?
260 PRINT"WOW - YOU MADE IT!"
270 PRINT "FUEL LEFT";F
280 END

290 PRINT "WHOOPS. NO MORE FUEL."
300 IF V<0 THEN 330
310 PRINT "YOU WILL DRIFT FOREVER"
320 END

330 PRINT"YOU WILL COAST TO YOUR "
340 PRINT"MOTHERSHIP - BUT YOUR PARTNER "
350 PRINT "IS DONE FOR.":END

360 PRINT "TIME UP":GOTO 300
                    'subroutine to display parameters
370 PRINT:PRINT"TIME";T
380 PRINT"SPEED";V
390 PRINT"APPROACH SPEED";V-U
400 PRINT "RANGE";H-Y
410 PRINT "FUEL";F
420 PRINT:FOR DL=1 TO 100:NEXT DL         'delay so that you can
                                          'watch the display
430 RETURN
```

HINTS AND TIPS

Cassette Tapes

The very first problem you will hit is to do with the cassette tape interface. Despite the fact that we have deliberately kept the programs in this book short, once you have keyed them in you will want to keep them and not re-key them every time. So, the first thing to get working is the cassette recorder. Don't wait to do this until you have got a long program to CSAVE; then it will be too late. Key in a tiny program such as:

```
10 REM THIS IS A VERY LONG REMARK
20 REM THIS IS ANOTHER LONG REMARK
30 REM THIS IS THE LAST LONG REMARK
```

Now try saving this by setting the recorder to Record and typing in direct mode:

```
CSAVE "TEST"
```

"TEST" is the file name which will be given to the file. The tape recorder will whir away and you should get the message:

```
OK
```

Now, try reloading the program - rewind the recorder and type:

```
CLOAD
```

Notice that the file name need not be specified, and this command will load the first program encountered on tape. Whilst the Dragon is searching for the program you will see the letter "S" in the top left corner. When it has found some data you will see

"F" in the top left corner and the file name "TEST". This changes from "F" to an inverse "F" at roughly one second intervals as the data is read in and assuming that it loads correctly you get the message: OK

The message "?IO ERROR" is bad news. It means that something has gone wrong. The program which you had in memory before will have been lost because it is removed when the file name is found (when the "S" changes to "F"). The one on tape cannot be loaded and you have lost all your work!

Some reviewers have been driven to despair by this problem, but it can be simply overcome. You now see why we suggest a small program for testing purposes, that is not too great a labour to rekey and rekey again until you have got it right.

The first tip is to use a DIN to DIN connector instead of the multiple jack plugs supplied. This is because the signal level from the Dragon on Record is about 1.5 volts - this level is too high for some Automatic Volume Control circuits to absorb. When the signal is applied to the Auxiliary Input circuits, which are expecting a signal from, say, another tape recorder, it is better able to cope.

The second tip is to play back the tape recording on the speaker before attempting to load it back into the machine. The signal should sound clean and crisp and at an absolutely deafening volume. If you cannot get the volume high enough, you might need a more powerful cassette recorder. Try different volume settings, all on the high side. The correct setting is quite critical, a very small shift of the controls could make all the difference.

There can be a problem with the automatic control of the Cassette Motor via the Remote cable. We suggest that you disable this control altogether by not putting in the jack plug at all. This means that you will have to do all the work in pressing the controls for Record or Play or Rewind, but since there is no automatic rewind, this is in fact easier. Most cassette tapes have a blank bit at the beginning called the "leader" tape. You cannot record on this because it is not magnetic. So, be careful that this bit has gone past before attempting to record. You can wind it on by hand or you can type:

```
MOTOR ON
```

which will set the motor going and let you pass over the leader tape. If you count up about ten very slowly this is usually long enough to be sure that the leader tape has gone through. Recording will start immediately when the (Enter) key is pressed for a CSAVE command so keep to this sequence:

Key the command CSAVE "XXXX" (but don't press (Enter))
Start the Recorder
Count slowly to ten   (one and two and three and four ...)
Press (Enter)

As a final tip, always save programs twice over. When you get the OK message quickly type CSAVE"XXXX" again, whilst the recorder is still running. This doesn't take long to do and sometimes save your bacon, because, if the first copy won't load, the second one nearly always does. Never go for more than half an hour of typing without saving the programs out. It sometimes also helps to use two tapes alternately, this then gives you four copies of the program that you could go back to in case of dire necessity. Always label tapes as you take them out of the machine - felt tipped pens work quite nicely on plastic.

Randomise

There is no Randomise function given in the reference card, or in the manual, for setting the seed of the random number generator. In fact, if you specify:

```
X=RND(-n)
```

N is a negative number and is then taken as the seed for the random number generator. Try this program with and without the first line to see the difference.

```
10 X=RND(-2)
20 FOR Z=1 TO 9: PRINT RND(20);:NEXT
```

Without the first line, the result will be different every time that the program is run. When the first line is put in, the results will be the same every time that you run it.

This technique is a good idea for testing games. Put in a line at the beginning to ensure that your random numbers are always the same and you can test it much easier - it is also a good way of "stacking the deck" to win by knowing in advance what's going to happen, but you wouldn't do that, would you?

Mug-trapping

This is the not very elegant name usually given for data validation and error detection - it's much shorter but not so polite. You will notice that all our programs contain after input statements extra commands to make sure that the data is what we were expecting e.g.:

```
10 INPUT "GO ON - (Y/N)";A$
```

the player must answer YES, Y, NO or N so we check for this as follows:

```
20 IF INSTR("YN",LEFT$(A$,1))=0 THEN 10
```

The LEFT$ function makes sure that even if YES has been entered, we look only at the left hand character so "YES ALRIGHT","YERSS","Y", and "YES" will all be reduced to a single "Y". The use of the INSTR function to check for all the possible values is also very powerful. It could be used to check for multiple letter values e.g.:

```
10 IF INSTR("ABC/CDE/FGH/CAR/",A$)=0
```

The use of the oblique strokes in the statement is so that combinations like "CCD" don't give a match when they should not.

IF THEN ...ELSE

Beginners are often confused by the use of IF THEN .. ELSE. The rule is that all commands following an IF statement are only obeyed if the condition is true. For example:

```
10 FOR Z=1 TO 10:IF A$(Z) = "THING" THEN 100:NEXT
```

This loop will only be obeyed once if the condition is not true the first time and therefore the NEXT command on the same line will be ignored. If the condition is false execution always (except for ELSE, see below) resumes at the next line found. So, we should recode this as follows:

```
10 FOR Z=1 TO 10:IF A$(Z)="THING" THEN 100
20 NEXT
100 REM PROGRAM CONTINUES
```

This will then operate ten times as we imagined.

Suppose we code as e.g.:

```
10 IF A$=B$ THEN GOSUB 100:GOSUB 200:GOTO 50 ELSE GOSUB 300
40 REM NEXT COMMAND
```

Notice first that the last command dependent on the IF is GOTO 50 and this is not separated by a ":" from the ELSE following. An equivalent piece of code would go as follows:

```
10 IF A$<>B$ THEN 30
12 GOSUB 100        'obeyed if condition is true
14 GOSUB 200
16 GOTO 50
30 GOSUB 300        'obeyed if condition not true
40 REM NEXT COMMAND
```

In other languages, like CORAL, for instance, the control of IF sequences uses a construct of BEGIN ..... END so that all the commands between the two are part and parcel of the same thing. It looks like e.g.:

```
IF A=B THEN BEGIN
            TEST-END              (you can also use names instead
            DISPLAY               (of subroutine numbers!
            GOTO 5Ø
            END
       ELSE BEGIN
            P3ØØ
            END
```

This is much easier to understand, and perhaps one day BASIC will include such a construct.

Control Keys

Study carefully the control keys given on the reference card:

(shift)/<-	means clear the current line (very useful)
(shift)/Ø	means set reverse case (usually very annoying)
(shift)/@	means freeze all output to the screen

The (Shift)/Ø commands can be obtained accidentally, when reaching for the ":" button, so be careful because commands typed in reverse case are not recognised and give syntax errors.

The Pause command, (Shift)/@, is also very useful, but remember that repeating it will not cause output to resume. To resume, you need any other character but that.

PRINT Commands

When a print command is obeyed so as to cause a line feed to the next line, the whole of the remainder of that line is destroyed in the process. Try these two programs:

```
10 PRINT @20,"HELLO";
20 PRINT @0,"A";           '";" character
30 PRINT @432,""

10 PRINT @20,"HELLO";
20 PRINT @0,"A"            'no ";" character
30 PRINT @432,""
```

With the first one, you see the word A followed by HELLO on the top line. With the second one, you see A alone on the top line. This is because the omission of the ";" character in line 20 has caused the remainder of the line to be deleted, as it spaces forward to the next line position. This can be a very useful trick for clearing the first line, but it can also be very puzzling when you didn't want this to happen!

Try these as well:

```
10 X=5:Y=6:Z=9
20 PRINT X;Y;Z
30 PRINT X,Y,Z
```

You get:

```
 5  6  9
 5               6
 9
```

A space is always printed in front of a number as well as behind. If the separator of "," is used then you get an automatic tab of sixteen character positions, which takes you to the next line for the third number. In practice the ";" format is nearly always what you want.

VOCABULARY TESTER

We wanted to include in this book a program which used data files held on cassette tape. There are clearly a lot of business applications which require this sort of treatment, but we thought that a vocabulary tester would be more generally useful to everyone. Some suggestions are given at the end for modifying the general structure for other applications.

The first difficulty is that other languages have characters which are not present on the Dragon keyboard. We have designed this example for French but it could be adapted for other languages equally well by using different conventions for the missing characters. We use the conventions as follows:

cedilla	becomes	","	e.g. garc,on
circumflex	becomes	"^"	e.g. ho^pital
grave accent	becomes	">"	e.g. infide>le
acute accent	becomes	"<"	e.g. employe<

Notice that the special character follows the character on which it is supposed to be written. This looks a bit strange at first but you can get used to it. Just beware of getting too used to it and writing words like that in examination papers! The accents run in the right direction if you look only at the top half of the character. We could have used "/" for acute and "\" for grave (this is CHR$(92) which can be displayed on the screen but not input from the keyboard), but this seemed best.

The idea of the program is that you build up your vocabulary lists on tape. Each time you are given a new set of words to learn you put them into the machine. When you think you know the new words then you merge in the current tape so that the old words are added onto the back. You then have a cumulative tape that you can use for revision later.

We have set a maximum of 1ØØ words for each tape. This is not a limitation of the machine, which could hold up to say 8ØØ words at a time, but perhaps a practical limit on the number you would want to revise at one time. You can change this limit if you think it is too low. All the statements referring to 1ØØ or 99 should be changed e.g. lines 4Ø, 14Ø and so on.

When the program is run, the Menu screen appears, which looks like this:

```
                VOCABULARY

1=READ DATA TAPE
2=WRITE DATA TAPE
3=ADD TO CURRENT LIST
4=TEST ENGLISH/FRENCH
5=TEST FRENCH/ENGLISH
6=REVIEW/CHANGE DATA

CHOICE?
```

To begin with, you won't have any data tapes to read in so we need option three to "Add to current list". Key "3" and press (Enter).

```
ADD NEW WORDS                 SPACE FOR 1ØØ

CEDILLA IS ,
GRAVE IS >
ACUTE IS <
CIRCUMFLEX IS ^
ENGLISH WORD ? PIG
NOW FRENCH EQUIVALENT
GENDER FIRST THEN WORD
E.G.LE GARC,ON

LE COCHON
```

The parts underlined in the example above are what you enter in response to the questions. When you have entered all the words, press (Enter) with no data, when asked for the English word, and you will return to the Menu screen. As you enter new words, the "SPACE FOR" field will be updated to show the remaining space, i.e. it becomes 99, then 98 and so on.

Next, you will want to learn the words and check that they are entered correctly so take option 6 to review or change the data. You see:

```
PIG/LE COCHON
Q=QUIT:Y=CHANGE ?
```

Press (enter) to go on to the next word. When you have been through them all, you will return to the menu screen again. If you type "Q" or "QUIT" then you will return to the Menu screen. If you type "Y" or "YES" you will be asked:

```
NEW VALUE ?    you enter the new English and French in the format
               displayed above e.g.  PIG/LE COCHON
```

This facility for amending the data is relatively crude, since you have to go right through all the words to get to the one you wish to change. If, however, you make a practice of checking and learning all the new words as you put them in and before you add them to the stored data, then this will be quite adequate.

Next we write out the data tape. For this take option 2. If you have the remote stop/start facility enabled (see hints and tips), then make sure that your tape is positioned after the leader tape and that play and record keys are down. The program will ask:

```
DATA TAPE READY ?
```

As soon as you press (Enter) the writing of the data tape will start. You will hear the relay inside clicking as different blocks of data are written onto the tape. When it returns to the

Menu screen, the operation is finished and you can rewind and remove the data tape. Don't forget to label it, and also to put on the number of words that it now contains.

To test yourself you can either have the machine ask for the English or for the French. Take either option four or five, as appropriate. Here is a sample dialogue:

```
ENGLISH BREAK
FRENCH CASSER
ENGLISH WHAT A SURPRISE
FRENCH QUEL SURPRISE          (got one wrong!
THE ANSWER IS QUELLE SURPRISE
ENGLISH TERRACE
FRENCH LA TERRASSE
ENGLISH SOMETIMES
FRENCH (Enter)                (had enough, so quit
SCORE 2 OUT OF 3
OK ?  (Enter)
```

and it returns to the Menu screen.

Your answers above are underlined. The testing part works by generating a random number within the range of items stored in memory and giving you one half of the stored string according to which way you are running the test. If you have a small number of items stored, then it might ask the same thing twice running, because the same random number comes up again - if this becomes annoying, then add some more words to the list.

Now we can merge on the rest of the data tape stored from previous occasions. To do this take option one. If there are any words in store, this will ask:

```
ADD TO EXISTING DATA ? Y
DATA TAPE READY ? (Enter)
```

Reading will start when (Enter) is pressed after "Data Tape Ready?". If you have said "N" for No then the existing words are removed and the new ones take their place. If when reading in the new items the array becomes full, then the message "FULL ?" appears. After you press (Enter), the remainder of the words on the tape are ignored. It is important for this reason to keep track of how many words are stored on your tapes. Only merge into a tape with room to take the new words.

To write out the amalgamated tape with all the words on together, take option 2 to write out the tape.

Lines to alter for more words per tape are marked '**

```
10 REM VOCABULARY TESTER
20 REM ,=CEDILLA;^=CIRCUMFLEX
30 REM >=GRAVE ACCENT;<=ACUTE
40 CLEAR 12000:DIM A$(100):N=0        '**
50 GOSUB 730   'MENU
60 ON MN GOTO 70,170,240,410,520,630    'option according
                                        'to menu choice
70 REM READ DATA TAPE
80 CLS1
90 IF N>0 THEN INPUT"ADD TO EXISTING DATA";A$   'merge question
100 IF LEFT$(A$,1)<>"Y" THEN FOR Z=1 TO 100:A$(Z)="":NEXT:N=0
110 INPUT"DATA TAPE READY";A$
120 OPEN"I",#-1,"VOCAB"              'open the data file
130 IF EOF(-1) THEN CLOSE:GOTO 50   'test for end of file
140 N=N+1:IF N>99 THEN INPUT"FULL ";A$:GOTO 50     '**
150 INPUT #-1,A$(N)                  'get a data record
160 GOTO 130

170 REM WRITE DATA TAPE
180 CLS1:INPUT"DATA TAPE READY";A$       'wait for tape ok
190 OPEN"O",#-1,"VOCAB"          'open output file
200 FOR Z=1 TO N
210 PRINT #-1,A$(Z)              'write a data item
220 NEXT Z
230 CLOSE:GOTO 50
```

```
240 REM ADD TO LIST
250 CLS1:PRINT"ADD NEW WORDS";TAB(18);"SPACE FOR";1ØØ-N;   '**
26Ø PRINT:PRINT"CEDILLA IS ,"
27Ø PRINT "GRAVE IS >"
28Ø PRINT "ACUTE IS <"
29Ø PRINT "CIRCUMFLEX IS ^"
3ØØ LINEINPUT"ENGLISH WORD ";A$
31Ø IF A$="" THEN 5Ø                    'end of entries
32Ø PRINT "NOW FRENCH EQUIVALENT"
33Ø PRINT"GENDER FIRST THEN WORD"
34Ø PRINT"E.G. LE GARC,ON"
35Ø PRINT:LINEINPUT B$
36Ø IF B$="" THEN 32Ø             'if english present then
                                 'french must be also
37Ø A$=A$+"/"+B$                 'form for storage
38Ø N=N+1:IF N>1ØØ THEN N=N-1:
        PRINT"SORRY - THIS SECTION IS FULL":GOTO 5Ø   '**
39Ø A$(N)=A$
4ØØ GOTO 24Ø

41Ø REM TEST ENGLISH/FRENCH
42Ø S=Ø:CLS1:G=Ø
43Ø X=RND(N)                      'get string for testing
44Ø GOSUB 84Ø                     'split the  string into two
45Ø PRINT "ENGLISH ";X1$
46Ø LINEINPUT"FRENCH ";Z$
47Ø IF Z$="" THEN PRINT "SCORE";S;"OUT OF ";G:
        INPUT "OK";Z$:GOTO 5Ø    'null means giving up
48Ø G=G+1                        'number of questions
49Ø IF INSTR(Z$,X2$)<>Ø THEN 51Ø  '<>Ø is OK
5ØØ PRINT "THE ANSWER IS ";X3$;" ";X2$:GOTO 43Ø
51Ø S=S+1:GOTO 43Ø               'good reply add one to score

52Ø REM TEST FRENCH/ENGLISH
53Ø S=Ø:CLS1:G=Ø
54Ø X=RND(N)
55Ø GOSUB 84Ø                    'split the string
56Ø PRINT "FRENCH ";X3$;" ";X2$
57Ø LINEINPUT"ENGLISH ";A$       'get the answer
```

```
580 IF A$="" THEN PRINT"SCORE ";S;" OUT OF ";G:
        INPUT"OK";A$:GOTO 50          'null means quit
590 G=G+1
600 IF INSTR(A$,X1$)<>0 THEN 620    '<>0 is OK
610 PRINT"THE ANSWER IS ";X1$:GOTO 540
620 S=S+1:GOTO 540                 'add one to score

630 REM REVIEW THE LIST/CHANGE
640 CLS1:S=1
650 IF S>N THEN 50                 'S>N means all through
660 PRINT A$(S)
670 INPUT "Q=QUIT:Y=CHANGE";A$
680 IF LEFT$(A$,1)="Y" THEN 690    'Y means change is required
685 IF LEFT$(A$,1)="Q" THEN 50     'Q means go back to menu
688 GOTO 710
690 INPUT "NEW VALUE";A$(S)
710 S=S+1:GOTO 650                 'step on for next item
720 END

730 REM DISPLAY MENU
740 CLS1:PRINT TAB(8);"VOCABULARY"
750 PRINT:PRINT"1=READ DATA TAPE"
760 PRINT "2=WRITE DATA TAPE"
770 PRINT "3=ADD TO CURRENT LIST"
780 PRINT"4=TEST ENGLISH/FRENCH"
790 PRINT "5=TEST FRENCH/ENGLISH"
800 PRINT"6=REVIEW/CHANGE DATA"
810 PRINT:INPUT"CHOICE";MN
820 IF MN<1 OR MN>6 THEN 740
830 RETURN

840 REM GET STRING AND SPLIT
850 X1$="":X2$=""
860 X$=A$(X):W=INSTR(X$,"/")  'find position of "/" character
870 IF W=0 THEN RETURN
880 X1$=MID$(X$,1,W-1):X2$=MID$(X$,W+1,LEN(X$)-W)
890 RETURN
```

The INSTR technique, used in this program for testing the replies, we have used before in the QUIZ program at the beginning. If verbs for instance are put as e.g.:

```
RECEVOIR
RECEIVE
```

If you answer, "To receive" instead of just "receive", then this will still be marked correct. Notice that if you fail to put in the gender, when answering in French, this will be marked wrong, which is quite right and proper.

This type of update program could be used for all sorts of different things. You could use it, for instance, to keep details of your gramophone record collection. To do this you might delete the testing part, but add a search facility to find any records by Beethoven or some specific title. The search bit would be coded as follows:

```
1000 INPUT "SEARCH KEY";A$
1010 FOR Z=1 TO N        'N=no of stored records
1020 IF INSTR(A$(N),A$)<>0 THEN PRINT A$(N)
1030 NEXT Z
```

The front half of the string, which we now use for the English, could be the record title and the back half, which we now use for the French, could be the storage reference code.

You would need of course to make other modifications, but the mechanism of reading tapes, writing tapes and adding to the data would stay the same.

The whole business of keeping records on files, is one of the most powerful things that computers can do. It will work best when you have a random access file or disk unit. This is because you then need not bother with limits imposed by the machine's memory. Now, if you put all the machine's Pontoon winnings in a special money box, then perhaps .....

ENIGMA VARIATIONS

Probably the Enigma coding machine used by the German Army during the Second World War was responsible for the development of the electronic computer. The story of the work done at Bletchley Park has been told many times (see for instance "Most Secret War" by R.V. Jones - Hodder and Stoughton/Coronet Books). Much is still shrouded in mystery but without doubt Alan Turing was pre-eminently responsible for the design of the "Bombe" which the first computer was called.

For those who have never seen them it is difficult to imagine the sheer bulk of the early computers. I once attended a sherry party held inside the ACE computer at the National Physical Laboratory. Also, so the story goes, a group of important overseas visitors were being shown round the DEUCE computer. The salesman confidently strode over to the echo chamber which was used as the dynamic RAM for the machine and threw open the door. Inside, the astonished visitors found an engineer eating his sandwiches in peace and quiet, or so he thought, sitting on the loudspeaker enclosures inside the chamber.

All of which is a sneaky introduction to the following encoding program, which imitates the action of an Enigma machine and produces a secret code that is extremely difficult to crack unless you have an IBM 3Ø31 machine with a few days spare time.

The program relies upon the tip we have described earlier for setting the seed of the random number generator. Random numbers produced from a given seed are not truly random but repeat a predetermined sequence every time. So, there are three parts to the code. The first is the secret number used to unlock the key today. This can be anything up to 9 digits in length. The second is the order of the letters in the coding string. We have chosen below to put the letters as "ABCDEF...." followed by the numbers and special characters, but you could have them in any order that you have agreed with your control. This allows you to have 42!(factorial) permutations for your code (a monstrous number). Finally the encipherment is carried out by adding a random number to obtain the substitute letter to be used. This means that "AAA" for example would not be encoded as the same letter every time - counting the letter frequency will not help in cracking the code.

```
20 INPUT"KEY";A:IF A<0 THEN 20
30 A=-A:X=RND(A)
40 INPUT "ENCODE/DECODE";A$
50 IF A$="" THEN END       'null means quit
60 A$=LEFT$(A$,1)
70 IF A$="D" THEN S=1:PRINT"DECODING" ELSE S=0:PRINT"CODING"
80 REM CHANGE STRING W$ TO SUIT YOUR OWN CODE
90 W$="ABCDEFGHIJKLMNOPQRSTUVWXYZ 0123456789$#.,?":W=LEN(W$)
100 INPUT"ENTER TEXT";A$:IF A$="" THEN END
110 A=LEN(A$)
120 FOR N=1 TO A
130 X$=MID$(A$,N,1)
140 GOSUB 230        'find the letter position within w$
150 IF S=1 THEN 200 'decoding
160 Y=RND(W)+X
170 IF Y>W THEN Y=Y-W:GOTO 170
180 PRINT MID$(W$,Y,1);
190 NEXT N:PRINT:GOTO 100
200 Y=X-RND(W)       'decoding
210 IF Y<1 THEN Y=Y+W:GOTO 210
220 GOTO 180
230 X=INSTR(W$,X$)
240 IF X=0 THEN X=W
250 RETURN
```

COMPUTERMIND

In this version of the well known traditional game you are asked to guess at the combination of colours that the computer has chosen. You are allowed 15 tries, and each time the computer will tell you the number of correct colours in the right position and the number of correct colours in the wrong position.

There are four levels of difficulty which give four different colours, any four colours, five different colours, or any five colours, to be guessed.

```
10 PRINT TAB(10);"MASTERMIND"
20 CLS1
30 Z$="RGBWYO"
40 PRINT "YOU MAY CHOOSE FROM ANY OF"
50 PRINT "THESE COLOURS ";Z$
60 PRINT "ENTER YOUR GUESS AS A 4 OR 5"
70 PRINT "LETTER WORD E.G. RGBY, AND "
80 PRINT "PRESS ENTER.  I WILL TELL YOU"
90 PRINT "YOUR SCORE AS A NUMBER OF BLACK"
100 PRINT "(CORRECT COLOUR AND CORRECT "
110 PRINT "POSITION), AND WHITE (CORRECT "
120 PRINT "COLOUR BUT WRONG POSITION)."
130 PRINT "IF I FIND YOU ARE IN TROUBLE I"
140 PRINT "WILL LET YOU KNOW. ENTER "
150 PRINT CHR$(34);"QUIT";CHR$(34);" IF YOU WANT TO GIVE IN."
160 PRINT "PRESS ENTER TO START"
170 PRINT
180 INPUT X$
190 CLS1
200 INPUT "HOW COMPLEX ";A
210 CLS1:IF A>0 AND A<5 THEN 240
220 PRINT A;" IS NOT ALLOWED"
```

```
230 GOTO 200
240 N=4:IF A>2 THEN N=5
250 S=(A=INT(A/2)*2)
260 IF S THEN PRINT "ANY ";
270 PRINTUSING"#";N;
280 IF NOT S THEN PRINT " DIFFERENT";
290 PRINT " FROM ";Z$
300 GU$="":FOR A=1 TO N       'set up the string to be guessed
310 T$=MID$(Z$,RND(6),1)
320 IF S OR A=1 THEN 340      'S true means can be similar
330 IF INSTR(GU$,T$)<>0 THEN 310
340 GU$=GU$+T$:NEXT A
350 PRINT @23,"      ":PRINT @23,"";:INPUT G$
360 IF G$="QUIT" THEN 540
370 P=(G(0)+1)*32
380 IF P>480 THEN P=480
390 PRINT @P,G$;
400 IF LEN(G$)<>N THEN 550
410 B=0:W=0                   'reset counters
420 FOR A=1 TO N              'count blacks and whites
430 IF MID$(GU$,A,1)=MID$(G$,A,1) THEN B=B+1:GOTO 450
440 IF INSTR(GU$,MID$(G$,A,1))<>0 THEN W=W+1
450 NEXT A
460 PRINT B;" BLACK",W;" WHITE"
470 G(0)=G(0)+1
480 IF B=N THEN 570           'wins
490 IF G(0)>14 THEN 530       'loses
500 IF G(0)=14 THEN PRINT @499,"LAST CHANCE!";
          :PRINT @480, " ";:GOTO 350
510 IF G(0)>11 THEN PRINT"I THINK YOU ARE LOSING"
520 GOTO 350
530 PRINT "SORRY - YOU LOSE"
540 PRINT "THE ANSWER WAS ";GU$:GOTO 580
550 PRINT @P,"IS NOT VALID - TRY AGAIN"
560 GOTO 350
570 PRINT "YOU WIN"
580 PRINT "YOU TOOK ";G(0);"TURNS"
590 INPUT "ANOTHER GO";Z$
600 IF LEFT$(Z$,1)="Y" THEN RUN
```

REVERSE

For this game, you are given a string of numbers and you can reverse any number counting from the left hand end. The object is to re-arrange them into numerical sequence in the smallest number of moves. When you have the trick of it, it is very easy to amaze your friends. I won't spoil it by telling the secret.

```
10 CLS1
20 PRINT TAB(10);"REVERSE"
30 PRINT
40 PRINT "THE OBJECT OF THE GAME IS TO"
50 PRINT "RE-ARRANGE 9 DIGITS BACK INTO "
60 PRINT "SEQUENCE. YOU TELL ME HOW MANY "
70 PRINT"DIGITS TO REVERSE IN EACH TURN"
80 PRINT "(STARTING FROM THE LEFT). E.G."
90 PRINT "IF YOU WERE GIVEN:"
100 PRINT "5 6 7 8 9 4  3 2 1"
110 PRINT "NOW REVERSE 5 TO GIVE:"
120 PRINT"9 8 7 6 5 4 3 2 1"
130 PRINT "NOW REVERSE 9 TO WIN"
140 PRINT
150 INPUT "PRESS ENTER TO CONTINUE";Y$
160 DIM A(9)                'array for numbers
170 FOR I=1 TO 9
180 N=RND(9)
190 IF I=1 THEN 230
200 FOR J=1 TO I-1
210 IF N=A(J) THEN 180
220 NEXT J
230 A(I)=N
240 NEXT I
250 C=0         'counter of tries
260 CLS1
```

```
270 FOR I=1 TO 9     'print the new string
280 PRINT A(I);
290 NEXT I
300 PRINT:PRINT
310 INPUT "NUMBER TO REVERSE";N
320 IF N>0 AND N<10 THEN 350
330 PRINT "INVALID - TRY AGAIN"
340 GOTO 270
350 K=INT((N+1)/2)  'K=count loop for reverse
360 FOR I=1 TO K
370 J=A(I)          'J is work location
380 A(I)=A(N+1-I)   'take one out
390 A(N+1-I)=J      'put one back
400 NEXT I
410 C=C+1
420 FOR I=1 TO 9     'test if now in order
430 IF NOT I=A(I) THEN 260
440 NEXT I
450 FOR I=1 TO 9:PRINT A(I);:NEXT:PRINT
460 PRINT "YOU TOOK";C;" TURNS"
470 INPUT "ANOTHER GAME";Y$
480 IF LEFT$(Y$,1)="Y" THEN 170
490 END
```

LUCKY DIP

Morse Code Tester

The main problem in learning Morse code is not learning to send the code, but learning to receive it. To practice receiving you need an expert assistant to help. The Dragon can be this assistant and give you practice at different speeds with completely consistent sending, with the same "fist" every time.

You still need a helper to key in the message but the helper need know nothing of Morse code.

```
10 REM MORSE CODE SENDER
20 CLS1:PRINT TAB(8);"MORSE CODE SENDER"
30 INPUT"SPEED";SP         'higher numbers go slower
40 DS=INT(SP):DD=3*DS
50 GOSUB 230               'set up the code array
60 INPUT A$                'text to be sent
70 FOR L=1 TO LEN(A$)
80 W=ASC(MID$(A$,L,1))     'get next letter
                           'numbers
90 IF W>=48 AND W<=57 THEN W=W-48:GOSUB 150:GOTO 120
                           'letters
100 IF W>=65 AND W<=123 THEN W=W-55:GOSUB 150:GOTO 120
110 FOR DL=1 TO SP*100:NEXT DL      'inter word gap
120 FOR DL=1 TO SP*60:NEXT DL       'inter letter gap
130 NEXT L
140 GOTO 60
```

```
        'subroutine to "send" the code
150 S=M(W+1)
160 IF S=0 THEN RETURN
170 C=S-INT(S/10)*10
180 IF C=1 THEN SOUND 100,DS:GOTO 200
190 IF C=2 THEN SOUND 100,DD
                   'get next bit and inter-bit gap
200 S=INT(S/10):FOR X=1 TO 30*SP:NEXT X
210 GOTO 160
220 RETURN
                   'set up morse code array
230 DIM M(36)
240 DATA 22222,22221,22211,22111,21111,
        11111,11112,11122,11222,12222
250 DATA 21,1112,1212,112,1,1211,122,1111,11,2221,
        212,1121,22,12,222,1221,2122,121,111,2,
        211,2111,221,2112,2212,1122
260 FOR X=1 TO 36:READ M(X):NEXT X
270 RETURN
```

Standard Deviation

The standard deviation is a measure of the variability of statistical observations. Its main use is in predicting the limits between which all occurrences of the data are likely to lie, if the data follows a "normal" distribution. For example, 99.5 % will lie within plus or minus three standard deviations from the average. The program shown below will work even if the data includes 0 or negative numbers.

```
10 REM STANDARD DEVIATION
20 PT=Ø:SS=Ø:SM=Ø:CLS1
30 INPUT"ENTER DATA";A$
40 IF A$="" THEN 8Ø 'end is signified by (Enter)
50 A=VAL(A$):PT=PT+1
60 SM=SM+A:SS=SS+A*A
70 GOTO 3Ø
80 AV=SM/PT:R=AV^2
90 R2=SS/PT-R:IF R2<Ø THEN R2=ABS(R2)
100 R2=SQR(R2)
110 PRINT "STD";R2
120 PRINT "AVERAGE";AV
```

Reaction Tester

How fast can you react to an emergency stop? Are you really fit to drive, or have you had one over the eight? Find out with this reaction speed tester. When the message "NOW" appears on the screen, press a key. Your reaction time is then given. Try pressing it before the off and it screams "CHEAT CHEAT". Times about .3 seconds are about average, for us oldies. Really sharp people can make it in less.

```
10 REM REACTION TESTER
20 CLS1:PRINT TAB(8);"REACTION TESTER":PRINT
30 PRINT"PRESS A KEY WHEN YOU SEE ";CHR$(34);"NOW";CHR$(34)
40 FOR D=1 TO RND(3ØØØ):NEXT D     'random delay before start
50 TIMER=Ø
60 IF INKEY$<>"" THEN PRINT@448,"CHEAT - CHEAT":GOTO 15Ø
70 PRINT @(3+RND(9))*32+RND(2Ø),"*** NOW ***";
80 IF INKEY$="" THEN 8Ø
90 T=TIMER/44
100 PRINT @448,"YOU TOOK ";
110 PRINTUSING"##.##";T;
120 PRINT "SECONDS"
130 IF T>3 THEN PRINT"WAKEY WAKEY"
140 IF T<.2 THEN PRINT"SHARP TODAY AREN'T WE"
150 INPUT"PRESS ENTER FOR ANOTHER GO";A$
160 RUN
```

VAT Calculator

Value Added Tax involves some very fiddly calculations in either adding it on to goods values or taking it out to work out the input tax. This program does both at the same time. It produces:

- the goods value (assuming the input value is tax inclusive)
- the tax value (assuming the input value is tax inclusive)
- the tax that would be payable if the input value were exclusive of tax

```
10 REM VAT CALCULATION PROGRAM
20 V=.15   'CURRENT RATE OF VAT
30 CLS1:PRINT TAB(4);"GOODS";TAB(14);"VAT INC";TAB(24);"VAT EXC"
40 INPUT"AMOUNT";A$
50 IF A$="" THEN END
60 A=VAL(A$)
70 B=A/(1+V):C=A-B:D=A*V
80 PRINTUSING"###,###.## ";B;C;D
90 GOTO 40
```

Anagram Solver

Crossword fans will love this program - it creates anagrams of any word entered. The results that come up are really to prompt you into seeing words that can be formed, since a 7 letter word has 5040 permutations, which would take too long to run through. If you see a word that looks promising then use (Shift)/@ to stop the display.

```
10 REM ANAGRAM SOLVER
20 CLS1:INPUT"WORD";A$
30 N=LEN(A$):DIM N(N)
40 W=INT(31/(N+1))
50 FOR A=1 TO W
60 GOSUB 140
70 FOR B=1 TO N
80 PRINT MID$(A$,N(B),1);
90 NEXT B
100 PRINT"/";
110 NEXT A
120 PRINT
130 GOTO 50
140 FOR Z=1 TO N:N(Z)=0:NEXT
150 FOR Z=1 TO N
160 R=RND(N):IF N(R)<>0 THEN 160
170 N(R)=Z:NEXT Z
180 RETURN
```

Metric/Imperial Conversion

Given a measurement in either metric or imperial units, this program converts to the other type of units. Within limits it has a degree of artificial intelligence. If you specify 1.2IN or 1.2INS or 1.2INCHES it is all the same.

Length	Weight	Capacity	Area
MM	GM	CC	Sq MM
CM	KG	LItres	Sq CM
MEtres	TONNes		Sq ME
KM			HEctares
			Sq KM
INches	OZ	PT	Sq IN
FT	LB	GAllons	Sq FT
Yards	TONS		Sq YArds
MIles			ACres
			Sq MIles

Here is a sample run:

```
ENTER NUMBER THEN UNITS ? 25.4 CM
0.83 FT
0.28 Y
ENTER NUMBER THEN UNITS ? 1 SQ METRE
1.20 SQUARE Y
0.00 AC
ENTER NUMBER THEN UNITS ? CMS 22
ENTER NUMBER FIRST
ENTER NUMBER THEN UNITS ? 22
WHAT ARE THE UNITS?
ENTER NUMBER THEN UNITS ? 22 PECKS
UNITS NOT FOUND
ENTER NUMBER THEN UNITS ? 5 LI
1.10 GA
```

```
10 SZ=40             'array size
20 DIM AA$(SZ),A(SZ)
30 DATA "MM","CM","ME","KM",""
40 DATA "GM","KG","TONN",""
50 DATA "CC","LI",,"MM","CM","ME","HE","KM","","",""
60 DATA "IN","F","Y","MI",,"OZ"
70 DATA "LB","TONS",,"PT","GA",""
80 DATA "IN","F","Y","AC","MI","","",""
90 FOR X=1 TO SZ:READ AA$(X):NEXT X
100 DATA 1,10,1000,1E+6,0,1,1000,1E+6,0,1,
          1000,0,1E-6,1E-4,1,1E+4
110 DATA 1E+6,0,0,0,25.4,304.8,914.4,
          1609344,0,28.35,453.6,1016064
120 DATA 0,568.26,4546.09,0,6.451E-4,.09290,.8361,4046.72
130 DATA 2589903.3,0,0,0
140 FOR X=1 TO SZ:READ A(X):NEXT X
          'main loop
150 PRINT"ENTER NUMBER THEN UNITS"
160 INPUT A$
170 A=VAL(A$):IF A=0 THEN PRINT "ENTER NUMBER FIRST":GOTO 150
```

```
         'find the first non-numeric character
180 FOR X=1 TO LEN(A$)
190 IF MID$(A$,X,1)>="A"THEN 220
200 NEXT X
210 PRINT "WHAT ARE THE UNITS?":GOTO 150
220 A$=MID$(A$,X)  'UNITS
230 IF INSTR(A$,"SQ")<>0 THEN M=13 ELSE M=1
240 FOR N=M TO 40
250 IF N=20 AND M=13 THEN N=32
260 IF AA$(N)="" THEN 280
270 IF INSTR(A$,AA$(N))<>0 THEN 300
280 NEXT N
290 PRINT "UNITS NOT FOUND":GOTO 150
300 M=N+20:IF M>40 THEN M=M-40
310 FOR X=1 TO 2
320 IF A(M)=0 THEN 380
330 Y=A*A(N)/A(M)
340 PRINT USING"######.##";Y;
350 IF ((M>12 AND M<18)OR(M>32 AND M<38))
         AND M<>16 AND M<>36THEN PRINT " SQUARE";
360 PRINT" ";AA$(M)
370 M=M+1
380 NEXT X
390 GOTO 150
```

The program works by using scaling factors. Suppose we have input 25.4 cms; this is first split into "25.4" and "CMS". This is not square measure (see line 230), so we search the array AA$ and stop on element 2 which contains "CM". Variable M becomes 22 (line 300) and we calculate:

```
25.4* 10/304.8  (element 22) = 0.83 FEET (element AA$(22))
25.4* 10/914.4  (element 23) = 0.28 Y    (element AA$(23))
```

If the units contain the letters "SQ" then the search is confined to the square measure part of the table. The zeros in the table are included so that if there is no relevant higher unit, it will not be printed, (see line 320).

Label Printing

If you are lucky enough to have a printer then you can use it to produce labels to stick on your notepaper. If you don't have one you can still work the program and look at the output on the screen, but the practical use of this is doubtful!

```
10 REM PROGRAM TO PRINT LABELS
20 PR=2              'set this as PR=Ø to have screen output
3Ø IF PR=2 THEN OPEN"O",#-PR,"OP"   'open the printer file
4Ø CLS1
5Ø LINEINPUT "NAME   ";A$(1)
6Ø FOR Z=1 TO 4            'get four lines of address
7Ø PRINT"LINE ";Z;
8Ø LINEINPUT A$(Z+1)
9Ø NEXT
1ØØ INPUT"HOW MANY LABELS";N  'number of labels
11Ø FOR Z=1 TO N           'number off loop
12Ø FOR A=1 TO 5           'number of lines
13Ø PRINT #-PR,A$(A)       'print a line
14Ø NEXT A
15Ø FOR B=1 TO 3           'space between labels
                           'alter to suit
16Ø PRINT #-PR,""
17Ø NEXT B
18Ø NEXT Z
19Ø IF PR=2 THEN CLOSE
```

The interesting thing about this program is that it can be fooled to divert output to the screen or to the printer as you require. See line 13Ø which if PR=2 would read:

```
13Ø PRINT #-2,A$(A)
```

whereas if you have reset PR=Ø then it reads:

```
13Ø PRINT #-Ø,A$(A)
```

This could be a useful trick where you have a program which might require screen output and might require printer output according to user choice.

Date Validation

A very common problem in programming concerns checking dates for validity against all the known calendar rules. We assume, being English, that the sequence will be Day, Month, Year but there can still be considerable variation within that e.g.:

28.11.82
2.11.82
2.2.82
2/2/82
2 2 82   and so on ....

```
10 REM DATE VALIDATION
20 LINEINPUT "DATE ";A$  'get the date which might include ,
30 N=1:W=1
40 FOR Z=1 TO LEN(A$)     'check each character
50 T=ASC(MID$(A$,Z,1))    'wait until non-numeric character found
60 IF T<48 OR T>57 THEN D(N)=VAL(MID$(A$,W,Z-W)):N=N+1:W=Z+1
70 NEXT Z:D(N)=VAL(MID$(A$,W))       'last one is year
          '30 days hath September, April, June, and November
80 DATA 31,28,31,30,31,30,31,31,30,31,30,31
90 IF D(2)<1 OR D(2)>12 THEN PRINT "MONTH ERROR":GOTO 20
100 RESTORE
110 FOR Z=1 TO D(2):READ L:NEXT
      'check for leap year in February
120 IF D(2)=2 AND (INT(D(3)/4)*4=D(3)) THEN L=29
130 IF D(1)<1 OR D(1)>L THEN PRINT "DAY ERROR":GOTO 20
140 IF D(3)<1 THEN PRINT "YEAR ERROR":GOTO 20
      'print it out properly formatted
150 FOR Z=1 TO 3:PRINTUSING"## ";D(Z);:NEXT
160 PRINT" OK":GOTO 20
```

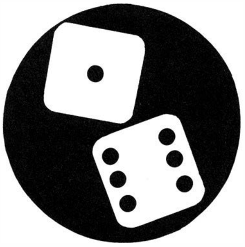
Memory Hex Display

A useful routine for peeking into the BASIC interpreter ROM to see what's going on. The list of valid commands starts at address 8Ø33 (hex) or 32819 (decimal). List it first in alphabetic form to see the command words - the only surprise is the command DLOAD, which presumably refers to disk file loading.

```
1ØØØ CLS1
1Ø1Ø PRINT "LIST MEMORY"
1Ø2Ø PRINT "USE @/SHIFT TO PAUSE"
1Ø3Ø PRINT "USE SPACE TO RESUME"
1Ø4Ø PRINT "BREAK TO STOP"
1Ø5Ø INPUT "ENTER MEMORY START ADDRESS (DEC)";F
1Ø6Ø INPUT "HEX(H)/ALPHA(A)";A$
1Ø7Ø IF INSTR("HA",A$)=Ø THEN 1Ø6Ø
1Ø8Ø H=(A$="H")
1Ø9Ø PRINT RIGHT$("ØØØØ"+HEX$(F),4);" ";
11ØØ FOR X=1 TO 8
111Ø Z=PEEK(F)
112Ø IF H THEN 115Ø
113Ø IF Z>32 AND Z<127 THEN PRINT CHR$(Z);ELSE PRINT ".";
114Ø GOTO 117Ø
115Ø Z$=RIGHT$("ØØ"+HEX$(Z),2)
116Ø PRINT Z$;" ";
117Ø F=F+1
118Ø NEXT X
119Ø PRINT:GOTO 1Ø9Ø
```

GRAPHICS SECTION

ROLLER DICE



Our family have been known to cheat, when in sticky situations, needing say a four desperately to avoid hotels on "Park Lane". Somehow, by accident the dice roll under the side-board and whoever retrieves them can say whatever they like! Nobody believes the convenient answer and violent arguments ensue. Use the computer to simulate the dice and you solve the problem. Anybody rolling the computer under the table is disqualified in a big way.

Whilst this example is a fairly easy one to follow, it provides the starting point for a detailed exposition of the graphics features of the Dragon. The four chapters following also deal with the same subject and form a detailed guide to all the excellent features which the Dragon provides. The whole subject is rather confusingly treated in the Basic instruction manual and hopefully a few worked examples of the facilities will clear the air.

The first thing to appreciate is that low level graphics and high level graphics are quite different things. They deal with different areas of memory and have different commands. the only commands of any relevance to low-level graphics are:

CLS c   Set whole screen to colour specified (c).

SET (x,y,c) Set a quarter character point, specified by coordinates x(across the screen) and y (down the screen) to colour c. The top left corner of the screen is Ø,Ø and bottom right of the screen is 63,31. This is a different numbering system to that used for PRINT @ and quite different from that used for high resolution graphics - very confusing.

RESET (x,y) Set a quarter character point to the background colour. Points x and y defined as before.

POINT (x,y) Function which returns Ø if the point is off, -1 if the point contains text and the colour if it is set.

SCREEN (Ø,n) Command used to reset the low resolution screen from high level graphics. In theory, should also be able to set the background colour for the above commands, but in practice has little effect.

Some of the effects produced by these commands are not quite what you might expect. Consider, for example, the following little program:

```
1Ø CLS1                'clear screen to green
2Ø SET(2Ø,2Ø,5)        'set a point to colour Buff
3Ø RESET(1Ø,1Ø)        'reset a point to background colour
4Ø CL=32: PRINT @383,CHR$(134+CL) ' print a  chequer pattern
                                   'character in colour blue
```

Run as it stands, it has some surprising effects. Firstly the SET command has set the whole square to the colour and not the point only. Secondly, the RESET has reset the point to Black and not Green as might have been expected. Thirdly, the background of the diamond character has been set to black, whereas green might have been expected.

Try changing line 1Ø to CLS. This time, the SET command has set only the single square, as requested, but has turned the remainder of the square to black! The RESET command this time has reset the whole square to black and not just the point. The CHR$ function produces the same effect.

Try also adding:

```
15 SCREEN Ø,Ø
```

or:

```
15 SCREEN Ø,1
```

You would expect these to have some effect, but they do not.

These limitations and problems can all be avoided if you clear the screen to black at the beginning by CLSØ. With a black background all these problems can be avoided and everything has the effect you might expect. The only problem is that every time the computer prints on the screen the character background reverts to green.

So, at last, we come to the example program:

```
1Ø REM DICE PROGRAM
2Ø CLSØ:CL=5                  'note screen cleared to black
                              'to try different colours
                              'change CL= statement
3Ø FOR P=11 TO 38 STEP 27     'P values of 11 and 38 are
                              'top left corners of the dice
4Ø TT=RND(6)                  'get the value for the dice
5Ø GOSUB 11Ø   'DRAW BOX
6Ø GOSUB 21Ø   'PAINT SPOTS
7Ø NEXT P
8Ø PRINT @48Ø,"";             'pause - note the ";" to prevent
                              'the whole line going green
9Ø INPUT A$                   'press (Enter) to roll again
1ØØ GOTO 2Ø
```

```
                                'first subroutine draws the square
110 REM DRAW BOX
120 Y=12:FOR X=P TO 15+P        'line across at the top
130 SET(X,Y,CL):NEXT
140 Y=24:FOR X=P TO 15+P       'bottom line across the screen
150 SET(X,Y,CL):NEXT
160 X=P:FOR Y=12 TO 24          'left hand vertical line
170 SET(X,Y,CL):NEXT
180 X=P+15:FOR Y=12 TO 24       'right hand vertical line
190 SET(X,Y,CL):NEXT
200 RETURN

                           'second subroutine fills in the spots
210 REM DRAW SPOTS
220 IF TT=1 OR TT=3 OR TT=5 THEN SET(P+8,18,CL) 'centre spot
230 IF TT=1 THEN RETURN
                      'top right corner and bottom left
240 IF TT=2 OR TT=3 THEN SET(P+12,15,CL):SET(P+4,21,CL):RETURN
250 SET(P+2,14,CL)  'four corner spots - top left
260 SET(P+13,14,CL) 'top right
270 SET(P+2,22,CL)  'bottom left
280 SET(P+13,22,CL) 'bottom right
290 IF TT<>6 THEN RETURN
300 SET(P+2,18,CL)  'middle left spot for a six
310 SET(P+13,18,CL) 'middle right spot for a six
320 RETURN
```

Try deleting lines 80 and 90. This will cause it to roll incessantly and provides a crude form of animation. It is unfortunately too slow to be of real use; the next chapters discuss more sophisticated methods of animation.

THE FLY AND THE TREACLE

We now move on to the higher level graphics, starting with the highest resolution (the most points). For this we use mode 4, which has two simple colour sets available, Black and Green (Set Ø) and Black and Buff (Set 1). Of these we are using Black and Green, so colour Ø is Black and colour 1 is Green. In this mode, only one bit of colour information is stored for each point (Ø or 1). Four pages of memory are required for this mode.

The program is very simple; a fly is flitting about in the forest above a pool of treacle which is conveniently handy. The fly moves about at random, going "buzz, buzz" in the plaintive and annoying way that flies have. Eventually he lands on the treacle and gets stuck. At this point he can't move any more and starts getting annoyed with a helpess "beep,beep". Just what he deserved! A silly program, but it illustrates some of the points about graphics.

First of all we must understand the effect of some of the special high resolution mode commands.

PCLEAR n Reserves n pages of graphics memory for use in storing data to be displayed. This does not clear the graphics area; it does not display the area; it can cause string variables to be erased, since graphics space would appear to be allocated above string space (for this reason, always put the PCLEAR command first in the program).

PMODE n,p Set the mode or resolution, which determine how information put into the graphics area will be recorded. N sets the mode and p sets the next page to be written into. In this mode there are four pages involved but p is the number of the first and by implication PMODE 4,1 defines pages 1 to 4 inclusive. This command does not clear the area or cause it to be displayed.

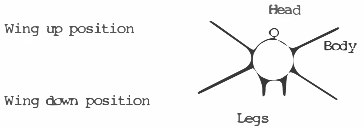
PCLS c   This command does initialise the area to the colour specified. By implication, it works for all the pages specified by the PMODE command. That is, if PMODE 4,1 has been set, then pages 1 to 4 inclusive will be reset by the command.

COLOR c1,c2   Foreground colour set as c1. Background colour set as c2. These terms have nothing to do with the border round the graphics square - they refer to the colour of INK and PAPER (to borrow terminology from elsewhere). LINE commands refer to the foreground colour as do PSET commands; PRESET commands set back to background colour. The range of colours available to you is always one less than those in the range - with a little thought you can see why; green writing on green paper is good for invisible writing but not much use for anything else. If you have more than two colours available, then neither foreground nor background need be the same colour as the initial value set by PCIS.

SCREEN 1,n   This is the vital command that causes a switch from the text mode (Ø) to the high level graphics mode (1). The current colour set is defined by n. In level 4 this can only be Ø or 1 and we are using set Ø. The page to be displayed is taken from the current setting of PMODE at the time that the command is obeyed. If, subsequently, PMODE page is changed, this does not cause the picture on the screen to alter. Only a subsequent screen command will cause the new setting to be displayed. This means that you can work on another page whilst displaying a totally different one. Executing an END statement causes an automatic switch back to mode Ø (text), so whilst working on program development it is best to include a statement e.g., 2ØØ GOTO 2ØØ. This causes an endless loop whilst you examine the picture. To get back again, use Break key. If the colour set is changed, by a SCREEN command from that in which it was initially written, the colours will change automatically, being re-interpreted in the new sense. We use this effect later, to produce a flashing sign board, with the colours displayed alternately.

The remaining commands we will introduce by example in the program. Remember that there are 256 elements across and 192 elements down the screen. Top left corner is Ø,Ø and bottom right corner is 255,191.

The picture on the screen has a green background for the forest, and a black rectangle at the bottom for the treacle. This is labelled "TREACLE" in case anybody should be in any doubt. Here is an enlarged picture of the fly.



```
1Ø REM FLY PROGRAM
2Ø PCLEAR 4             'reserve 4 pages of memory for mode 4
3Ø PMODE 4,1            'set highest definition and set page 1-4
                        'for writing into
4Ø PCLS1                'initialise pages 1-4 as colour Green
5Ø COLOR Ø,1            'set colour of INK as Black
                        'and PAPER as green
6Ø LINE (Ø,16Ø)-(255,191),PSET,BF  'draw a rectangle of black at
                        'the bottom of the screen (top left is
                        'Ø,16Ø; bottom right is 255,191
                        ',BF means draw a rectangle not a line
                        'and fill it with foreground colour (black)
```

```
70 A$="BM160,172;C1;R4;L2;D8;B;R4;U8;R4;D4;L4;F4;B;R2;R4;
     L4;U8;R4;B;D4;L4;D4;B;R6;U8;R4;D8;U4;L4;D4;B;R10;L4;
     U8;R4;B;R2;D8;R4;B;R6;L4;U8;R4;B;D4;L4;"
                        'draw the word TREACLE in green inside
                        'the treacle area at the bottom
                        'BM is a blank move C1 sets colour green
                        'letter 'T' e.g. is Right 4,Left 2,Down
                        '8,Blank move, Right 4; then ready for
                        'next letter
80 DRAW A$
90 COLOR 0,1            'reset colour because the DRAW command
                        'has used Green Ink and reset foreground
                        'colour
100 X=10+RND(235):Y=7+RND(140)      'starting position - constants
                              'ensure that it is all on screen
110 D=0:GOSUB 220 'DRAW
120 GOSUB 320                'flap the wings
130 SOUND 255,1:SOUND 255,1  'go "buzz,buzz"
140 D=1:GOSUB 220  'UNDRAW
150 DX=9-RND(18):DY=11-RND(18) 'DX and DY are the intervals
                             'by which fly moves to new position
                             'note DY has a positive bias to
                             'make sure he comes to earth
                             'eventually.
160 X=X+DX:IF X<10 THEN X=10 'make sure all of it is still
                             'on the screen
170 IF X>245 THEN X=245
180 Y=Y+DY:IF Y>154 THEN 420 'if Y>154 then the fly is stuck
                             'he can dip his toes in but
                             'no more!
190 IF Y<7 THEN Y=7
200 SCREEN 1,0               'the vital display command
210 GOTO 110

220 REM DRAW/UNDRAW FLY
230 CIRCLE(X,Y),3,D          'CIRCLE for head at x,y
                             'Radius 3 Colour D
                             'if D=0 (Black) it draws against
                             'green background
```

```
                                     'if D=1 (Green) it undraws against
                                     'green background
240 PSET (X,Y-3,D)                   'set a single point as the head
250 COLOR D,1-D                      'set foreground colour for
                                     'Draw /Undraw as above
260 LINE(X-2,Y+3)-(X-2,Y+6),PSET    'draw the left leg
270 LINE(X+2,Y+3)-(X+2,Y+6),PSET'   'draw the right leg
280 RETURN

320 REM WINGS UP/DOWN
330 LINE(X-4,Y-1)-(X-10,Y-6),PSET  'draw left wing up
340 LINE(X+4,Y-1)-(X+10,Y-6),PSET  'draw right wing up
350 LINE(X-4,Y-1)-(X-10,Y+6),PSET  'draw left wing down
360 LINE(X+4,Y-1)-(X+10,Y+6),PSET  'draw right wing down
370 LINE(X-4,Y-1)-(X-10,Y-6),PRESET     'undraw left wing up
380 LINE(X+4,Y-1)-(X+10,Y-6),PRESET     'undraw right wing up
390 LINE(X-4,Y-1)-(X-10,Y+6),PRESET     'undraw left wing down
400 LINE(X+4,Y-1)-(X+10,Y+6),PRESET     'undraw right wing down
                                    'note that at end no wings are
                                    'showing at all
410 RETURN
                              'enter here when fly is stuck
                              'first draw the word "HELP"
420 A$="BM100,172;C1;D8;B;R4;U8;D4;L4;D4;B;R10;L4;
         U8;R4;B;D4;L4;D4;B;R10;L4;U8;B;R6;B;D8;U8;R4;D4;L4;"
430 DRAW A$
440 COLOR 0,1             'reset colour because DRAW was in green
450 SOUND 255,4:SOUND 200,4    'yelp of distress
460 D=0:GOSUB 220:GOSUB 320    'draw and flap wings helplessly
470 GOTO 450
```

In the animation sequence, where the wings flap, the wings are drawn first as up, then down, then the up is removed, then the down is removed. This gives the effect of flapping, since for an instant you can see both positions simultaneously. You might like to try the effect of re-arranging them in a different sequence.

TRAIN PICTURE

This time a slightly more colourful picture. An old fashioned train puffs along the line and disappears into the tunnel. We use mode 1, which gives a wider range of colours and allows the maximum number of pages to be used for graphics. We also do the animation by a more sophisticated method, which is more suitable for cartoon type programs. This allows characters to appear in front of or behind objects as they move across the screen.

First the new commands:

PCOPY n1 TO n2 This command makes a straight copy of a single page of graphics memory. It does only copy a single page. So for mode 1 graphics which take 2 pages per screen it is necessary to execute two separate copy commands. Copying does not affect the screen being displayed (unless of course that is the screen being copied to).

PAINT (x,y),c2,c2 In operation this command can be tricky. In concept it is simple - it is as though you spilled a pot of paint of colour c1 at a given spot and it spreads until it hits a barrier, which is the defined colour spots of colour c2. It follows from this that the x,y coordinates must be within the area which you are trying to paint. Spill your paint outside the barrier and it fills the whole screen, which can be disconcerting, to say the least. Suppose you have drawn a circle in red (colour 4), on a blue background (colour 3) and you are trying to paint it red. Clearly the barrier colour cannot be blue or the painting process will stop immediately when it hits the blue left inside the circle. So it must be red. Now if the coordinate references fall actually on the spots forming the circle you have just drawn, then it will stop immediately having found the red barrier. You must be careful that the coordinate references fall inside the blue interior of the circle. This is best achieved by specifying the same references as for the middle of the circle.

GET (x,y)-(x2,y2),an,G To save a particular part of a picture, use this command. It "gets" the part of the picture specified by x,y (top left corner) and x2,y2 (bottom right corner) from the current display screen, and stores it in the array an. Array an must have been dimensioned previously at the right size to hold this amount of data, one array element per spot on the screen. The array dimensions will be x2-x and y2-y respectively. If parameter G is specified then the action code (see below) **must** be specified for PUT.

PUT (x,y)-(x2,y2),an,PSET To do the reverse of GET and put back the parts of the picture, use PUT. Because it needn't be put back in the same place as it was in originally, this is a way of producing motion in the picture and is the basis for the train program. There are other action codes besides PSET, but they are rather complicated. If a picture is PUT in a position where part will fall off the screen, strange things happen!

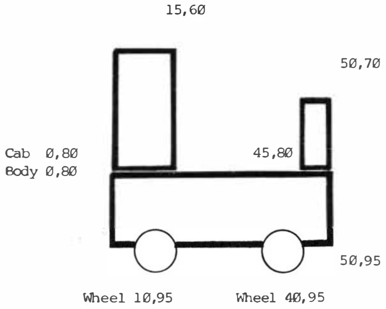
We shall be using mode 1, which has a resolution of 128 X 96 spots on the screen. Logically one might expect that the spots would be numbered so that bottom right corner was 127,95 but in fact it keeps the same numbering as we had before with mode 4 and bottom right is 255,191 again, as before.

```
10 PCLEAR 6          'we use three set of pages in groups of 2
20 DIM P(50,40)                 'array dimensioned to take
                                'the train picture from 0,60
                                'to 50,100
```

```
30 PMODE 1,1                             'PMODE 1 set with page 1,2
                                         'to write into initially
                                         'colour set for this mode is:
                                         '1=Green, 2=Yellow, 3=Blue,
                                         '4=Red
40 PCLS3                                 'Initialise to colour blue
                                         'representing the sky
50 COLOR 2,4                      'foreground yellow background red
60 LINE(0,100)-(255,191),PSET,BF  'draw a rectangle at the
                               'bottom of the screen colour yellow
                               'representing the ground
70 CIRCLE (220,30),10,2        'circle drawn in yellow, radius 10
80 PAINT(220,30),2,2           'paint the circle yellow as sun
90 PCOPY 1 TO 3:PCOPY 2 TO 4   'copy the basic picture to page 3,4
100 PCOPY 1 TO 5:PCOPY 2 TO 6  'and to 5,6
                               'now draw in the train
110 LINE(0,80)-(50,95),PSET,B  'draw in yellow the outline of
                               'the body as a rectangle
120 PAINT (5,85),4,2           'paint the inside but not the
                               'border itself red
130 LINE (0,80)-(15,60),PSET,B      'draw another rectangle
                               'in yellow as the cab
140 PAINT (10,70),4,2          'paint red leaving yellow border
150 LINE(50,70)-(45,80),PSET,B      'draw funnel
160 PAINT (46,79),4,2          'paint the inside red
170 CIRCLE (10,95),5,4         'draw the rear wheel
180 PAINT (10,95),4,4          'paint the whole wheel red
190 CIRCLE(40,95),5,4          'draw the front wheel
200 PAINT(40,95),4,4           'paint the whole wheel red
210 GET(0,60)-(50,100),P,G     'save this part of the picture
                               'into array P
220 PCOPY 3 TO 1:PCOPY 4 TO 2  'put back the original picture
                               'into page 1,2
230 FOR Y=0 TO 200 STEP 4      'start of movement sequence
240 FOR X=3 TO 5 STEP 2        'use alternately page 3,4:5,6
250 PCOPY 1 TO X:PCOPY 2 TO X+1     'copy back the basic picture
260 PMODE 1,X                  'set new page to write into
270 IL=INT(X/5)*2              'half a step value
280 PUT(Y+IL,60)-(50+Y+IL,100),P,PSET   'put back the train
```

```
                                        'but a little further on than
                                        'last time
281 COLOR 2,4                           'reset colour because PAINT'
                                        'command has changed the setting
282 LINE(200,50)-(254,98),PSET',BF  'draw the tunnel on the right
                                        'this will overwrite part of the
                                        'train if it has passed into
                                        'the tunnel
290 SCREEN 1,0                          'display the new picture
300 NEXT X                              'next page
310 NEXT Y                              'next step
320 SOUND 200,8:SOUND 150,8             'end of the line, blow whistle
330 GOTO 230                            'repeat
```

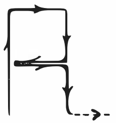
The foreground (yellow) the background (blue sky) with the sun at top right can be easily imagined, but here is a detailed picture of what the train should look like:



SIGN WRITING

From time to time, there is a need to print text on the high resolution graphics screens. The fly and the treacle program had two examples of this, where we wanted to print a word on the picture. To save re-inventing this every time, here is a standard subroutine for writing anything you like. It produces an alphabetical string for each letter, so the result should always be prefaced by commands to move to the desired position, set the colour and set the scale. The basic characters are designed as 16 points down and 8 points across, with a separation between letters of four points. So, if you said, for example, scale was 8, this would be double the basic units (4 is the reference level - you can draw to a smaller scale or to a larger scale) and these would become 32 units down, 16 units across, and 8 units separation.

Each character is assumed to start at its bottom left corner, and the string finishes at the bottom right corner, ready for the next letter to be added. Here is an example of the plotting for the letter A:



The example program, shown below, uses this subroutine to draw an illuminated sign of a person's name. Ideally the name should be about 6 characters long, and can have a space after it to make the whole thing look prettier. The name is then plotted down the screen in different scales. The actual scales to be used are given in the DATA statement, so you can change it if you want all big letters, or all small letters, for example. The colours are each plotted in turn. So the sequence goes yellow, blue, red against a green background and then repeats. If the whole of the name won't fit onto a line then it gets on as many letters as it can.

At the end, the whole screen is displayed in the alternative character set, which is Green, Purple, Magenta, Cyan against a Buff background; these colours alternate at about 3 second intervals.

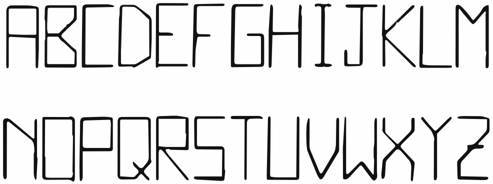
```
10 REM EYEBALL CHARACTERS
20 PMODE 1,1         'set mode 1 and page 1 to write into
30 PCLEAR 2:CLEAR 2000   'clear 2 pages and string space
40 PCLS1             'initialise the screen area
50 INPUT "NAME";A$:IF A$="" THEN END    'get the name string
60 GOSUB 290  'SET UP ARRAY
70 SCREEN 1,0     'display screen  so that the draw sequence
                  'can be observed
                  'DATA statement (scale, point size) repeated
80 DATA 3,12,4,16,6,24,7,28,8,32,10,40
90 READ S:READ PT:X=0:Y=0:W=LEN(A$):IL=3
100 Y=Y+PT+IL        'next line Y reference IL is the
                     'inter-line gap
110 FOR Z=2 TO 4 'COLOURS
120 FOR A=1 TO LEN(A$)
130 GOSUB 590 'GET LETTER STRING
140 IF X+(.75*PT)>255 THEN 190    'no more will fit on
150 DRAW "BM"+STR$(X)+","+STR$(Y)+";C"+STR$(Z)+";S"
                +STR$(S)+";"+Z$
160 X=X+.75*PT       'move x to next letter position
170 NEXT A           'next letter
180 NEXT Z:GOTO 110      'next colour - repeat when colours done
190 REM NEW LINE
200 IF Y>170 THEN 240          'reached bottom of screen
210 X=0:READ S:READ PT         'get next scale
220 IF S>3 THEN IL=5           'adjust gap
230 GOTO 100
```

```
             'all lines drawn
240 SCREEN 1,0            'first colour set
250 FOR DL=1 TO 1000:NEXT DL  'wait
260 SCREEN 1,1            'second colour set
270 FOR DL=1 TO 1000:NEXT DL  'wait
280 GOTO 240                 'repeat

        'initialise the string arrays
290 REM SET UP ARRAY
300 DIM Z$(26)
310 Z$(0)="B;R20;"                              'space
320 Z$(1)="U16;R8;D8;L8;R8;D8;B;R4;"            'A
330 Z$(2)="U16;R8;D4;G4;L4;R4;F4;D4;L8;B;R12;"      'B
340 Z$(3)="U16;R8;B;D16;L8;B;R12;"              'C
350 Z$(4)="U16;R4;F4;D8;G4;L4;B;R12;"           'D
360 Z$(5)="U8;R4;L4;U8;R8;B;D16;L8;B;R12;"      'E
370 Z$(6)="U8;R4;L4;U8;R8;B;D16;B;R4;"          'F
380 Z$(7)="U16;R8;B;D8;L4;R4;D8;L8;B;R12;"      'G
390 Z$(8)="U8;R8;L8;U8;B;R8;D16;B;R4;"          'H
400 Z$(9)="B;R2;R4;L2;U16;R2;L4;B;R10;B;D16;"       'I
410 Z$(10)="U4;B;U12;R8;L4;D16;L4;B;R12;"       'J
420 Z$(11)="U16;D8;E8;G8;F8;B;R4;"              'K
430 Z$(12)="U16;D16;R8;B;R4;"                   'L
440 Z$(13)="U16;F4;E4;D16;B;R4;"                'M
450 Z$(14)="U16;F8;U8;D16;B;R4;"                'N
460 Z$(15)="U16;R8;D16;L8;B;R12;"               'O
470 Z$(16)="U16;R8;D8;L8;B;R12;B;D8;"                 'P
480 Z$(17)="U16;R8;D12;G4;L4;B;R8;H4;F4;B;R4;"        'Q
490 Z$(18)="U16;R8;D8;L8;F8;B;R4;"                    'R
500 Z$(19)="R8;U8;L8;U8;R8;B;D16;B;R4;"               'S
510 Z$(20)="B;R4;U16;L4;R8;B;R4;B;D16;"               'T
520 Z$(21)="U16;B;R8;D16;L8;B;R12;"                   'U
530 Z$(22)="B;U16;D12;F4;E4;U12;B;R4;B;D16;"          'V
540 Z$(23)="U16;B;R8;D16;H4;G4;B;R12;"                'W
550 Z$(24)="U4;E4;H4;U4;B;R8;D4;G4;F4;D4;B;R4;"       'X
560 Z$(25)="B;U16;F4;E4;G4;D12;B;R8;"                 'Y
570 Z$(26)="B;U16;R8;D4;G4;R2;L4;R2;G4;D4;R8;B;R4;"     'Z
580 RETURN
```

```
59Ø REM MAKE UP DRAW STRING
6ØØ P=ASC(MID$(A$,A,1))
61Ø IF P<65 OR P>91 THEN P=Ø:GOTO 63Ø
62Ø P=P-64
63Ø Z$=Z$(P)
64Ø RETURN
```

If you want to use this subroutine, start by loading it into memory, renumber it to say line 2ØØØ and carry on writing the program. When you have finished, save the whole thing onto a separate tape.

This is the full character set - notice that there are problems with diagonal strokes. Luckily, a diagonal stroke e.g. "E4" actually produces the equivalent of "U4" and "R4" instead of about 5.6 which would be the actual length of the hypotenuse.



CASTLE WALLS

The Normans have arrived! Standing on top of the parapet, you can look down and see the ladders starting to rise up the walls - but what can you do?

BOILING OIL! you think, and immediately start to pour the oil down onto the invaders. Sometimes you miss, sometimes you hear the sounds as the victims writhe in agony.

It is inevitable that they'll get in sooner or later, it's just a question of how long you can keep them out. Oil is in plentiful supply, but once your cask empties (watch the dipstick at the right of the screen) that's all they need to rush in, whilst you run helplessly to and fro. You pour the oil by using the "V" key, wait until the man on the parapets is over the top of a ladder before pressing it.

Initially you can select the level of difficulty - the more difficult, the more ladders start coming but the better the score for each victim. Once the ladders reach the top your score is shown.

```
10 REM CASTLE WALLS
20 REM (C) 1982 PHIPPS ASSOCIATES
30 PCLEAR 2:PMODE 1,1                   'set graphics mode
40 GOSUB 570                            'initialise
50 GOSUB 490                            'instructions
60 INPUT "WHAT LEVEL (1-5)";DV
70 IF DV<1 OR DV>5 THEN 60
80 ND=DV+1:DG=234/ND                    'calculate ladder gap
90 R1=.2+DV/10                          'ladder growth rate
100 R2=5+DV*2                           'oil stopping factor
```

```
110 R3=(5-DV)*10                              'man movement speed
120 G=1000:S=0                                'initial oil and score
130 GOSUB 610                                 'draw the castle
140 DP=20:RP=235:DIR=1                        'set left & right limits
150 DIM D(ND,2)                               'ladder position array
160 Y=16+DG/2                                 'first ladder X-coord
170 FOR X=1 TO ND                             'position ladder bases
180 D(X,1)=INT(Y+.5) AND &HFFE:D(X,2)=175
190 Y=Y+DG
200 NEXT X
210 FOR P=DP TO RP STEP DIR*8                 'move man
220 C=FG:GOSUB 940                            'draw him
230 IF INKEY$="V" THEN GOSUB 710              'check if oil poured
250 IF RND(0)>R1 THEN 330                     'see if ladder growth
260 IF INKEY$<>"" THEN 260                    'ensure key not pressed
270 D=RND(ND):B=D(D,2):A=D(D,1)               'select ladder to grow
280 DRAW "C4;BM"+STR$(A)+","+STR$(B)+";U7;R1;D2;R5;U2;
         R1;D7;L1;U4;L5;D4"                   'ladder graphic pattern
290 SOUND 176-B,1                             'rising crescendo
300 IF B<18 THEN 380                          'see if ladder at top
310 D(D,2)=B-8                                'set new ladder height
330 C=BG:GOSUB 940                            '"undraw" man
340 NEXT P
350 LET DIR=DIR*-1                            'change man's direction
360 XP=DP:DP=RP:RP=XP                         'swap castle edges
370 GOTO 210                                  'repeat movement

380 REM GAME END
390 SCREEN 0,0 : CLS
400 PRINT @96,"YOUR SCORE THAT TIME WAS";S
410 IF S<10 THEN PRINT "DISGUSTING."
420 IF S>=10 AND S<=20 THEN PRINT "...ONLY FAIR."
430 IF S>20 AND S<40 THEN PRINT "PASSABLE..."
440 IF S>=40 AND S<100 THEN PRINT "GETTING BETTER!"
450 IF S>99 AND S<200 THEN PRINT "FAIRLY GOOD!"
460 IF S>199 AND S<400 THEN PRINT "YOU'VE PLAYED THIS BEFORE!"
470 IF S>399 THEN PRINT "AN EXPERT!"
480 END
```

```
490 REM INSTRUCTIONS
500 CLS
510 PRINT "    **** CASTLE WALLS ****"
520 PRINT : PRINT " THE MARAUDING HORDES OF NORMANS
                 HAVE ARRIVED AT CASTLE EGO. YOU
                 HAVE 1000 GALLONS OF BOILING OIL
                 AT YOUR DISPOSAL; USE IT WISELY!"
530 PRINT " YOU CAN POUR THE OIL ONTO THE
          RISING LADDERS BY PRESSING THE
          'V' KEY."
540 PRINT " WHEN THE OIL HAS ALL BEEN USED,
          YOUR TOTAL SCORE WILL BE SHOWN.
          HOW HIGH CAN YOU MAKE IT?"
550 PRINT
560 RETURN

570 REM INITIALISE
580 BG=3:FG=1:COLOR FG,BG                'set colour scheme
590 M$="R5;D3;L2;D2;U2;L3;U2"            'define man's graphic
600 RETURN

610 REM DRAW CASTLE
620 PCLS 2
630 COLOR FG,BG
640 SCREEN 1,0
650 DRAW "BM0,0;C3;R16;D16;R224;U16;R15;D177;L255;U177"
660 PAINT (128,0),3,3 :PAINT (128,191),1,3
670 P=20:C=FG:GOSUB 940                  'position the man
680 COLOR 4,BG
690 LINE(248,175)-(248,74),PSET:COLOR FG,BG
700 RETURN

710 REM POUR OIL
720 IF G<1 THEN SOUND 1,1:RETURN         'no oil left
730 COLOR 3,2
740 FOR Z=1 TO ND:IF ABS(P-D(Z,1))<8 THEN 760     'find ladder
750 NEXT Z:Z=0                           'oil has missed
760 FOR Y=16 TO 168 STEP 8               'draw oil down screen
770 FOR W=0 TO 6 STEP 2                  '...in steps of 2
```

```
780 LINE (P,Y+W)-(P+6,Y+W),PSET
790 NEXT W
800 IF Z THEN IF Y+8>D(Z,2) THEN IF R2>RND(100) THEN 830
                                            'pour over ladder
810 NEXT Y
820 Y=Y-8
830 X=Y+7                                   'point to ladder rung
840 IF Z THEN FOR N=200 TO 175 STEP -1:SOUND N,1:NEXT N
                                            'screams of victims
850 FOR Y=18 TO X STEP 2                    'erase oil line
860 IF Z THEN LINE (P-7,Y)-(P+15,Y),PRESET ELSE
              LINE (P,Y)-(P+6,Y),PRESET
870 NEXT Y
880 IF Z=0 THEN 910                         'see if ladder was hit
890 S=S+1+INT((X-D(Z,2))*DV/8)              'yes - update score
900 D(Z,2)=X                                'reposition ladder
910 IF G<0 THEN G=0                         'remaining oil
920 G=G-20-RND(60+DV*20):
          LINE (248,74)-(248,175-INT(G/10)),PRESET
                                            'reduce oil and dipstick
930 RETURN

940 REM DRAW MAN
950 DRAW "BM"+STR$(P)+",10;C"+STR$(C)+";"+M$
960 RETURN
```

SNAKES AND LADDERS

A traditional game for two players. The non-traditional twist is that the snakes can bite and land you in hospital for three turns and you can also fall off the ladder and have to go back to square one for two turns!

The program draws the board at the top of the screen, using blue and yellow chequered squares. Square one is top left and square 100 is bottom left. Numbering goes in the traditional sequence, i.e. you go backwards on alternate rows. To follow the play properly you do need a colour television set, since the ladders are marked in red and the snakes in orange. You have to throw a six to start and you must end exactly on 100. If you go over 100 then it will count backwards for the excess.

It is a difficult program to test since a game takes so long to play - like the board game, the play seems to go on for ever. It can be made easier if the INPUT statement in line 810 is replaced by a PRINT statement and it will then play without human intervention.

A different board is generated for every game. See lines 100-200 below.

```
10 REM SNAKES AND LADDERS
20 DIM BD(100)
30 CLS1:PRINT TAB(6);"SNAKES AND LADDERS"
40 PRINT:PRINT"YOU MUST START ON A SIX"
50 PRINT"YOU GET AN EXTRA GO FOR A SIX"
60 PRINT"YOU MUST END EXACTLY ON 100"
70 PRINT
80 INPUT"FIRST PLAYER";PL$(0) 'get player names
90 INPUT"SECOND PLAYER";PL$(1)
```

```
100 REM SET UP THE BOARD
110 FOR Z=1 TO 100:BD(Z)=Ø:NEXT
120 FOR Z=1 TO 25
130 H=RND(99):IF BD(H)<>Ø THEN 130
140 H2=8Ø-RND(16Ø):IF H2+H<1 OR H2+H>99  THEN 14Ø
         'the snakes and ladders must end up still on board
         'the ends of the hazards are marked with 999 which
         'is later removed;  also a snake and a ladder cannot
         'join up and you cannot have a snake at square 1ØØ!
15Ø IF BD(H2+H)<>Ø THEN 14Ø
16Ø BD(H)=H2:BD(H+H2)=999
17Ø NEXT:CLS1
18Ø FOR Z=Ø TO 3:PP(Z)=1:NEXT
         'now take out the end markers
19Ø FOR Z=1 TO 1ØØ:IF BD(Z)=999 THEN BD(Z)=Ø
2ØØ NEXT
21Ø S(Ø)=Ø:S(1)=Ø    'turn indicators
22Ø GOSUB 64Ø        'draw the board
23Ø G=2-RND(2)    'decide who goes first Ø OR 1

24Ø REM MAIN LOOP
25Ø GOSUB 81Ø     'wait for throw and clear screen
                  'print name, position, throw
26Ø PRINT PL$(G);" IS ON ";PP(G);:PRINT" THROWS ":GOSUB 46Ø
27Ø T=RND(6):PRINT T;:GOSUB 46Ø            'delay
28Ø IF S(G)<-1 THEN 48Ø                    'missing a turn
29Ø IF S(G)=Ø AND T<>6 THEN PRINT"NOT STARTED YET":
        GOTO 48Ø ELSE S(G)=-1          'must start on a six
3ØØ PP(G)=PP(G)+T                          'update board position
31Ø PRINT " AND MOVES TO ";PP(G)
32Ø GOSUB 85Ø                              'update board display
33Ø IF PP(G)=1ØØ THEN 52Ø                  'wins
34Ø IF PP(G)<1ØØ THEN 37Ø                  'on board position
              'drop through if off the board
35Ø PRINT"OH DEAR TOO FAR - MOVING BACK":GOSUB 46Ø
36Ø PRINT "TO ";:PP(G)=2ØØ-PP(G):PRINT PP(G):GOSUB 85Ø
```

```
370 REM CHECK HAZARDS
380 IF BD(PP(G))=Ø THEN 44Ø           'no snake or ladder
                    'snake action
390 IF BD(PP(G)))<Ø THEN PRINT "HISS - HISS ":GOSUB 57Ø:
        PRINT"WHEE - DOWN WE GO TO ";:GOSUB 46Ø:
        PP(G)=PP(G)+BD(PP(G)):PRINT PP(G):GOSUB 85Ø:GOTO 38Ø
                    'ladder action
400 PRINT "OH LOOK! HERE'S A LADDER"
410 PRINT "UP WE GO TO ":GOSUB 46Ø:PP(G)=PP(G)+BD(PP(G)):
        PRINT PP(G):GOSUB 85Ø
420 IF RND(6)=1 THEN PRINT"OH DEAR - YOU FELL OFF ":
    PRINT "YOU MISS TWO TURNS AND GO TO 1":S(G)=-3:PP(G)=1:
    GOSUB 85Ø:GOTO 48Ø
430 GOTO 38Ø
                    'test for another go otherwise turn over
440 IF T=6 THEN GOSUB 82Ø:PRINT "ANOTHER GO":GOTO 24Ø
450 GOTO 48Ø


                    'delay subroutine
460 FOR DL=1 TO 3ØØ:NEXT DL
470 RETURN
                    'takes action for missed turns
480 REM CHANGE OVER PLAYERS
490 G=1-G       '1 becomes Ø and Ø becomes 1
                'test for missed turns
500 IF S(G)<-1 THEN S(G)=S(G)+1:GOSUB 82Ø:
        PRINT PL$(G);" MISSES A TURN ":GOTO 49Ø
510 GOTO 24Ø

520 REM PLAYER WINS
530 PRINT PL$(G);" WINS"
540 INPUT "ANOTHER GAME ";A$
550 IF LEFT$(A$,1)="Y" THEN GOTO 1ØØ
560 END
```

```
570 REM TEST FOR SNAKE BITE
580 IF RND(10)<>1 THEN RETURN
590 PRINT "AARGH! - SNAKE GOT YOU"
600 PRINT "SNAKE SERUM IS AT THE HOSPITAL"
610 PRINT"AT SQUARE 43 - MISS THREE TURNS"
620 S(G)=-4:PP(G)=43            '-4 means miss three turns
630 RETURN

640 REM SR TO DRAW BOARD        'snakes are orange
650 PP=0                        'ladders are red
660 FOR Y2=1 TO 3 STEP 2        'alternate squares are blue/yellow
670 FOR Y3=1 TO 25
680 DY=-1:GOSUB 760
690 NEXT Y3
700 FOR Y3=25 TO 1 STEP -1      'alternate rows go backwards
710 DY=0:GOSUB 760
720 NEXT Y3
730 NEXT Y2
740 PRINT @(Y2)*32,"";
750 RETURN
              'decide colour and print square
760 PP=PP+1:IF INT(PP/2)<>PP/2 THEN CS=16 ELSE CS=32
770 IF BD(PP)<0 THEN CS=112
780 IF BD(PP)>0 THEN CS=48
790 PRINT @(Y2+DY)*32+Y3+3,CHR$(143+CS);
800 RETURN
              'clear screen and wait for enter button
810 PRINT@448,"";:INPUT"THROW";A$
820 FOR Y=4 TO 14            'alternate entry to clear only
830 PRINT @Y*32,""
840 NEXT:PRINT@128,"";:RETURN
```

```
850 REM SR TO UPDATE BOARD
860 REM PP(G)=NEW PP(G+2)=OLD
870 IF PP(G+2)=<PP(G) THEN DY=1 ELSE DY=-1    'DY+=->; -=<-
880 FOR Y2=PP(G+2)+DY TO PP(G) STEP DY
890 BP=Y2-DY:GOSUB 1010
          'test to see if off the board
900 IF Y2>100 OR Y2-DY>100 THEN PC=0:GOTO 990
910 IF Y2-DY=PP(1-G) THEN PRINT@BP,MID$(PL$(1-G),1,1);:GOTO 960
920 IF INT(Y2/2)<>Y2/2 THEN PC=32 ELSE PC=16
930 IF BD(Y2-DY)<0 THEN PC=112      'snake
940 IF BD(Y2-DY)>0 THEN PC=48       'ladder
950 PRINT@BP,CHR$(143+PC);
960 BP=Y2:GOSUB 1010
970 PRINT @BP,MID$(PL$(G),1,1);
980 SOUND Y2,1                'tone rises as players go up the board
                              'tone falls when descending
990 NEXT Y2:PRINT@128,"";     'reset cursor below board display
1000 PP(G+2)=PP(G):RETURN     'update old position array
          'work out BP character position from board position
1010 BP=BP-1:B2=INT(BP/25):B1=BP-B2*25
1020 IF BP<0 THEN B1=0:B2=0
1030 IF B2/2<>INT(B2/2) THEN B1=24-B1
1040 BP=B2*32+B1+4
1050 RETURN
```

HUNT THE WUMPUS

The Wumpus is a mythical creature with Bat-like wings and sucker feet. This is a picture of what the artist thinks it looks like, but nobody who has seen one, has lived to tell the tale! He lives in a maze of inter-connected caves, through which you must travel to find him. Some of the caves contain bottomless pits; fall into one of these and you are never seen again. There are also two super bats, who have a playful habit of picking people up and dumping them in another cave.

The network of caves is generated each time in a different way - so making maps of the cave structure won't help you. You are provided with five arrows, with which to shoot the Wumpus. Beware, the arrow might come back and hit you on the rebound. If you run out of arrows, you also lose, so use them carefully.

When you are one cave away from a hazard, you will get a warning, for example:

I SMELL A WUMPUS	The wumpus is in one of the caves next to yours.
I FEEL A DRAUGHT	You are next to a bottomless pit.
BATS NEARBY	A cave next to yours contains a Super-Bat.

The Wumpus is normally a lazy creature, but should he land in your cave, well let's not dwell on the unimaginable, suffice it to say, that you lose.

The Wumpus will only move when he hears an arrow being fired, and he moves at random, to one of the caves connecting to his. When you fire an arrow, it is able to go into up to five caves. You will be asked how many caves, and the cave numbers that you want it to go into. If the caves do not connect by a tunnel, then the arrow may fly at random, and could well hit you, and you lose.

Good hunting!

```
10 CLS1:PRINT TAB(8);"HUNT THE WUMPUS"
20 PRINT:INPUT "DO YOU WANT THE RULES";Y$
30 IF LEFT$(Y$,1)="Y" THEN GOSUB 400     'print the rules
40 DIM H(5),C(25),R(5)    'H() is the location of hazards
                          'C() is the cave map
                          'R() are the target caves for shooting
50 PRINT "PLEASE WAIT A MOMENT"
60 FOR X=1 TO 25     'set up caves
70 N=RND(25)
80 IF X=1 THEN 120
90 FOR Y=1 TO X-1
100 IF N=C(Y) THEN 70
110 NEXT Y
120 C(X)=N
130 NEXT X
140 REM SET UP HAZARDS
          'h(Ø)=current cave position  - no cheating please
          'h(1)=wumpus position
          'h(2),(3) =bats position
          'h(4),(5)=pits position
150 FOR X=Ø TO 5
160 N=RND(25)
170 IF X=Ø THEN 210
180 FOR Y=Ø TO X-1
190 IF N=H(Y) THEN 160
200 NEXT Y
210 H(X)=N
220 NEXT X
230 A=5 'NO OF ARROWS
240 CLS1
```

```
250 REM MAIN CONTROL LOOP
260 S=H(Ø)
270 P=1:Q=Ø
280 GOSUB 68Ø                  'check and print hazards
290 PRINT "YOU ARE IN CAVE";C(H(Ø))
300 PRINT "TUNNELS LEAD TO ";
310 P=3:GOSUB 68Ø              'Print connecting caves
320 PRINT
330 PRINT "SHOOT OR MOVE ";
340 GOSUB 168Ø              'get a 1 char reply
350 P$=Y$
360 N=1
370 IF P$="S" THEN GOSUB 1Ø2Ø 'shoot subroutine
380 IF P$="M" THEN GOSUB 139Ø 'move subroutine
390 ON N GOTO 25Ø,17ØØ,175Ø   'main loop,win,lose

4ØØ CLS1    'RULES PRINT
41Ø PRINT "THE WUMPUS IS A CREATURE WITH"
42Ø PRINT "SUCKER FEET AND IS TOO HEAVY "
43Ø PRINT "FOR BATS TO LIFT.  HE LIVES IN"
44Ø PRINT "A NETWORK OF 25 CAVES, AND YOUR"
45Ø PRINT "TASK IS TO HUNT HIM DOWN AND "
46Ø PRINT "SHOOT HIM WITH ONE OF YOUR FIVE"
47Ø PRINT "ARROWS.  TWO OF THE CAVES ARE "
48Ø PRINT "BOTTOMLESS PITS - IF YOU FALL"
49Ø PRINT "IN, YOU LOSE. TWO OTHERS "
5ØØ PRINT "CONTAIN SUPERBATS WHICH WILL "
51Ø PRINT "WHISK YOU TO ANOTHER CAVE AT "
52Ø PRINT "RANDOM.":PRINT
53Ø GOSUB 67Ø             'pause for enter key
54Ø PRINT "WHEN YOU ARE ONE CAVE AWAY FROM"
55Ø PRINT "A HAZARD, I WILL GIVE YOU A "
56Ø PRINT "WARNING, E.G."
57Ø GOSUB 82Ø:GOSUB 83Ø:GOSUB 84Ø      'wumpus,bats,draught msg
58Ø PRINT"AN ARROW CAN GO IN UP TO 5 "
59Ø PRINT "CONNECTING CAVES. THE WUMPUS"
6ØØ PRINT "WAKES AND MOVES ONE CAVE WHEN "
61Ø PRINT "HE HEARS AN ARROW. YOU LOSE IF:"
62Ø PRINT "1:THE ARROW HITS YOU"
```

```
630 PRINT"2:THE WUMPUS GET'S IN YOUR CAVE"
640 PRINT "3:YOU RUN OUT OF ARROWS"
650 PRINT"4: YOU FALL IN A PIT"
660 PRINT"GOOD HUNTING"
670 INPUT"PRESS ENTER TO CONTINUE";Y$:CLS1:RETURN

680 REM CHECK AND PRINT HAZARDS
690 FOR X=1 TO 4
700 GOSUB 970
               'message,check cave, print connecting
710 IF L<>Ø THEN ON P GOSUB 770,950,740
720 NEXT X
730 RETURN
740 REM PRINT CONNECTING CAVES
750 PRINT C(L);
760 RETURN
               'Q can equal Ø or 5
770 REM CHOOSE MESSAGES
780 FOR Y=1 TO 5
790 IF L=H(Y) THEN ON INT(Y+Q) GOSUB 820,830,830,840,840,
                                      850,880,880,920,920
800 NEXT Y
810 RETURN
820 PRINT "I SMELL A WUMPUS":RETURN
830 PRINT"BATS NEARBY":RETURN
840 PRINT "I FEEL A DRAUGHT":RETURN
850 PRINT "WUMPUS GOT YOU"
860 N=2
870 RETURN
880 PRINT"ZAP-SUPERBAT SNATCH"
890 CS=RND(25)
900 F=-1
910 RETURN
920 PRINT"YIIIEEE - FELL IN PIT"
930 N=2
940 RETURN

950 IF CS=L THEN F=Ø
960 RETURN
```

```
970 K=S+((X=1)*5)-(X=2)-((X=3)*5)+(X=4)
980 L=Ø 'L REF IS CONNECTING CAVE
99Ø IF K<1 OR K>25 THEN RETURN
1ØØØ L=K
1Ø1Ø RETURN

1Ø2Ø IF A>Ø THEN 1Ø5Ø 'SHOOT SR
1Ø3Ø PRINT "OUT OF ARROWS"
1Ø4Ø GOTO 137Ø
1Ø5Ø INPUT"HOW MANY CAVES (1-5)";R
1Ø6Ø IF R<1 OR R>5 THEN 1Ø5Ø
1Ø7Ø FOR Z=1 TO R
1Ø8Ø PRINT"ENTER CAVE NO";Z;
1Ø9Ø INPUT R(Z)
11ØØ NEXT Z
111Ø S=H(Ø)
112Ø FOR Z=1 TO R          'test each target in turn
113Ø CN=R(Z)
114Ø F=-1
115Ø GOSUB 162Ø            'get element reference
116Ø P=2
117Ø GOSUB 68Ø
118Ø IF F THEN GOSUB 156Ø
119Ø IF CS=H(Ø) THEN 135Ø          'arrow is fatal to you
12ØØ IF CS=H(1) THEN 133Ø          'hit the wumpus
121Ø S=CS
122Ø NEXT Z
123Ø CLS1
124Ø PRINT "MISSED"
125Ø A=A-1           'now one less arrows
126Ø S=H(1)
127Ø GOSUB 156Ø 'MOVE WUMPUS
128Ø H(1)=CS
129Ø L=H(Ø)
13ØØ Q=5
131Ø GOSUB 77Ø       'choose messages
132Ø RETURN
```

```
1330 N=3
1340 RETURN

1350 CLS1
1360 PRINT"OUCH - ARRROW GOT YOU"
1370 N=2
1380 RETURN

1390 INPUT"WHERE TO";CN 'MOVE SR
1400 CLS1
1410 GOSUB 1620            'get array reference
1420 IF CS=0 THEN 1540     'invalid cave?
1430 F=-1
1440 P=2
1450 GOSUB 680
1460 IF F THEN 1540        'does it connect?
1470 L=CS
1480 F=0
1490 Q=5
1500 GOSUB 770             'messages
1510 H(0)=CS
1520 IF F THEN 1470
1530 RETURN
               'error processing
1540 PRINT"IMPOSSIBLE -"
1550 RETURN

1560 X=RND(4)
1570 GOSUB 970
1580 IF L=0 THEN 1560
1590 CS=L
1600 CN=C(2)
1610 RETURN
```

```
                'turn cave reference into array element no
162Ø FOR K=1 TO 25
163Ø IF CN=C(K) THEN 166Ø
164Ø NEXT K
165Ø K=Ø        'on return K=cave element no
166Ø CS=K
167Ø RETURN
                'get a one character reply
168Ø INPUT Y$:Y$=LEFT$(Y$,1)
169Ø RETURN
                'losing ending
17ØØ PRINT "YOU LOSE"
171Ø PRINT "PLAY AGAIN WITH SAME CAVES";
172Ø GOSUB 168Ø           'get 1 character reply
173Ø IF Y$="Y" THEN 14Ø
174Ø END
                'winning ending
175Ø PRINT "YOU GOT THE WUMPUS"
176Ø PRINT "YOU WIN"
177Ø GOTO 171Ø
```

PONTOON

You may have seen some programs which play Blackjack on the American model. Pontoon is very similar, but the European version has some important differences. So, perhaps a description of the game is in order first.

The object of the game is to draw a number of cards, which approach as closely as possible 21 in value, hence the name Vingt et Un in the European version. For this purpose, court cards, Jack, Queen, King count as ten and an Ace may count as 1 or as 11, as the holder wishes. Two cards, consisting of an Ace and a King, constitute a Pontoon, when these are the only two cards held, and possession of a Pontoon gets special rewards.

A banker (or dealer) is appointed and a card is dealt to each player, and bets are placed on that first card. At this point the dealer may "double" (i.e. require that all bets are doubled). A second card is dealt to each player. If the Bank now holds a Pontoon, it must declare it and all players pay the banker. Supposing that the Bank does not hold a pontoon, each player in turn is then asked whether he wishes to "Twist" (receive a card face up), "Buy" (receive a card face down for an increased stake), or "Stick" (receive no more cards). It is usually obligatory to take another card with a hand with a value of 15 or less. If a player holds two of a kind in his hand, he may "split" the hand and play it as two separate hands. Anyone holding five cards in total representing a score of 21 or less wins automatically as if he held a pontoon.

To simplify the game a little, I have invented special house rules. First, the computer is the banker and remains so, come what may. This is because the dealing is done by computer. Because the Bank will not change hands in the normal way, the pack is reshuffled whenever the Bank would have changed hands.

Second, "Buying" of cards is not allowed, this is because it is impossible to deal a card face down on the screen! Thirdly, "Splitting" is not allowed. This is quite straightforwardly to keep the program size within bounds. I felt it was more important to allow one or two players, than to allow splitting, which is a rarefied feature of the game. Lastly, "Sticking" on any number is permitted. This is because otherwise the game becomes very dull with the computer making all the decisions about dealing another card or not. You can if you wish play your own rules on this point.

The Bank's playing strategy is very simple. It will double, if dealt an Ace as its first card. This is because it then has a good change of getting a pontoon, with a court card or a ten as the next card. When deciding whether to add to its hand, the Bank first looks at the state of play; if all the players are bust, then it sticks on whatever it then holds. After this it will "Stick" if holding a best score of 16 or greater. It is possible to break the Bank by winning all the Bank's money, and it has been done, but is surprising how often this simple strategy succeeds against more adventurous opponents. The Bank's resources are not infinite to make it fair. The Bank is given the same amount of money as the wealthiest of the players.

The best strategy for winning is in fact to copy the Bank's example and stick on 16. It also pays to watch the pack sequence very carefully, because all returned cards are put onto the bottom of the pack. If you have memorised the sequence then you can predict the next card to come out. To encourage this practice, the deal subroutine prints a message "GOING THROUGH THE PACK AGAIN", when it restarts at the top.

The settlement of the bets is quite complex. This part of the program is given in the rule processing section (lines 7ØØ -76Ø). In words rather than program, they can be summarised as follows:

1. If the bank holds a natural pontoon, and the player does not, then the player pays twice his original stake.
2. If the bank holds a natural pontoon, and the player does also, then the player forfeits his original stake.
3. If the player has "bust" (holds a minimum score of > 21) then his stake is forfeit to the bank.
4. If the player holds five cards, and the bank does not, then the player receives twice his original stake.
5. If the player holds five cards, and the bank does as well, then the player receives an amount equal to his stake.
6. If the player holds a pontoon, then he receives twice his original stake.
7. If the player's best score is greater than the bank's best score then he receives an amount equal to his original stake. It will be greater automatically, if the Bank has bust.
8. If the player's best score is equal to or less than the bank's best score then his stake is forfeit to the bank.

These conditions are evaluated by the use of "Truth" variables set to -1 if true and to Ø if not true. These are so called because the statement e.g.:

```
IF TV THEN PRINT "TRUE" ELSE PRINT "FALSE"
```

will be obeyed as it reads if the variable TV has been set as described.

The key variables which are evaluated and set by subroutine 119Ø are:

	For Player	For Bank
Busted (min score >21)	BU	BU
Holding five cards	C5	B5
Holding Pontoon	PT	BP
Ordinary variables		
Best score	MT	BT
Minimum score	HT	HT

The variable WR is set to a positive value if the player is paying the bank and to a negative value if the bank is paying the player. For example WR=-2 means that the bank pays the player twice his original stake.

To add a bit of fun, I have included a random remarks generator, to give the computer a little more personality. At certain key events, starting a new hand, the bets, player going bust, losing a player and end of the game, the remarks subroutine is entered. 8Ø% of the time it says nothing, but the other 2Ø% it selects a remarks from those set up - about five different ones are provided for each event. You can substitute your own remarks, if you wish, and yours needn't be quite so polite. Our testers developed the theory that whenever the Bank says "I feel lucky today", it loses. Inspection of the program should reveal that this is an unlikely hypothesis, but it has yet to be disproved experimentally.

Since this is a complicated program, it may be difficult to get going. Some diagnostic hints might help. The pack is contained in array P. To print out the pack, type in direct mode:

```
FOR Z=1 to 52:PRINT P(Z);:NEXT
```

All the numbers should be different, and in the range 1-52. Variable P should be set between 1 and 52 and point to the next card to come out of the pack.

To look at players' hands, type:

```
FOR Z=1 TO 15:PRINT HD(Z);:NEXT
```

Player one's cards should be elements 1-5, player two's 6-1Ø, and the Bank's 11-15.

Also, look at array PG which contains for each player the number of cards in his hand:

```
FOR Z=1 to 3:PRINT PG(Z);:NEXT
```

If you think that the settling up of bets is wrong, then press Break and type:

```
PL=1 (or PL=2):GOSUB 119Ø:PRINT BU;C5;PT;HT;MT;AT;
```

The first three are the truth variables, followed by the lowest score, the best score, and the number of aces. Variable WR should be set to the result of the hand.

```
10 REM PONTOON VINGT ET UN
20 CLEAR 1000:DIM A$(25):GOSUB 1640            'set up remarks
         'P() is the pack;HS$() is the alpha equivalent of suit
         'HD$() is the alpha equivalent of hand cards
         'HS() is the suits of the cards held
         'HD() is the values of the cards held
         'PG() is the number of cards in each hand
30 DIM P(52),HS$(15),HD$(15),HS(15),HD(15),PG(3)
40 CLS1:PRINT TAB(5);"PONTOON":PRINT      'house rules display
50 PRINT "HOUSE RULES":PRINT
60 PRINT "(1) I AM THE BANKER"
70 PRINT "(2) NO SPLITS ALLOWED"
80 PRINT "(3) NO BUYING OF CARDS"
90 PRINT"(4) YOU MAY STICK ON ANY NUMBER"
100 INPUT"WHAT ARE THE MAXIMUM STAKES";MX      'get maximum bets
110 INPUT"HOW MANY PLAYERS(BESIDES ME)";H      'no of players
120 IF H<1 OR H>2 THEN 110
130 FOR Z=1 TO H                 'get number of chips held
140 PRINT "PLAYER";Z;:INPUT "NAME";PL$(Z)
150 INPUT "HOW MUCH MONEY HAVE YOU";M(Z)
160 IF M(Z)<1 OR M(Z)>9999 THEN
     PRINT"I DON'T BELIEVE YOU":GOTO 150
170 NEXT Z
                   'give the bank the larger of the two
180 IF M(1)>M(2) THEN M(3)=M(1) ELSE M(3)=M(2)
190 PRINT "I HAVE";M(3)
200 GOSUB 900  'SHUFFLE PACK
210 REM MAIN ENTRY
               'look at state of money for each player
220 GOSUB 1330  'WHO IS STILL IN
230 IF ABS(WR)=2 THEN GOSUB 900     'reshuffle the pack if
                                'pontoon or five card hand
                                'reset the variable arrays
240 CLS1:FOR Z=1 TO 15:HS$(Z)="":HD$(Z)="":HS(Z)=0:HD(Z)=0:NEXT
250 FOR Z=1 TO 3:B(Z)=0:PG(Z)=0:NEXT
                   'print out the chip totals
260 PRINT M(1);TAB(11);"BANK";M(3);TAB(26);M(2)
270 PL=3:GOSUB 1040:GOSUB 1100      'deal two cards to bank
280 GOSUB 1040:GOSUB 1100           'without timing delay
```

```
290 PRINT PL$(1);TAB(16);PL$(2)     'print names at top
300 PRINT
310 R=1:GOSUB 1580              'remarks for starting a hand
320 FOR Y=1 TO H                'deal a card to each player
                                'and get bets from each
330 PL=Y:TB=15*(Y-1):GOSUB 1020:GOSUB 1160:GOSUB 1100
340 PRINT TAB(TB);:INPUT"YOUR BET";B(Y)
350 IF B(Y)<1 OR B(Y)>MX THEN 340
360 IF B(Y)>M(Y) THEN PRINT TAB(TB);"NO IOUS!":GOTO 340
370 R=2:GOSUB 1580          'remarks for placing of bets
380 NEXT Y
                            'bank doubles if Ace is first card
390 IF HD(11)=11 THEN PRINT"I DOUBLE YOU":FOR Y=1 TO H:
            B(Y)=B(Y)*2:NEXT Y
400 FOR Y=1 TO H:PL=Y           'deal second card for each
410 TB=(Y-1)*15:GOSUB 1020:GOSUB 1160:GOSUB 1100
420 NEXT Y
                            'bank declares pontoon if held
430 PL=3:GOSUB 1190:IF NOT PT THEN 490
440 PRINT"I HAVE A NATURAL PONTOON"
450 FOR Z=11 TO 12
460 PRINT HD$(Z);" OF ";HS$(Z)
470 NEXT Z
480 BP=-1:GOTO 680
                        'enter here if bank does not hold pontoon
490 FOR Y=1 TO H:TB=(Y-1)*15
500 PL=Y:GOSUB 1190
510 IF PG(PL)>=5 THEN 560       'cannot hold > 5 cards
520 PRINT TAB(TB);:INPUT"TWIST/STICK";R$       'get decision
530 R$=LEFT$(R$,1):IF R$<>"T" AND R$<>"S" THEN 510
                                'if twist then deal another card
540 IF R$="T" THEN GOSUB 1020:GOSUB 1160:GOSUB 1100:
        GOSUB 1190:IF  NOT BU THEN 500
                                'print busted message
550 IF R$<>"S" THEN PRINT TAB(TB);"YOU BUSTED":R=3:GOSUB 1580
560 NEXT Y:TB=0
570 GOSUB 1540:PRINT "I HOLD" 'declare the banks hand
580 FOR Z=11 TO 12
590 PRINT HD$(Z);" OF ";HS$(Z)
```

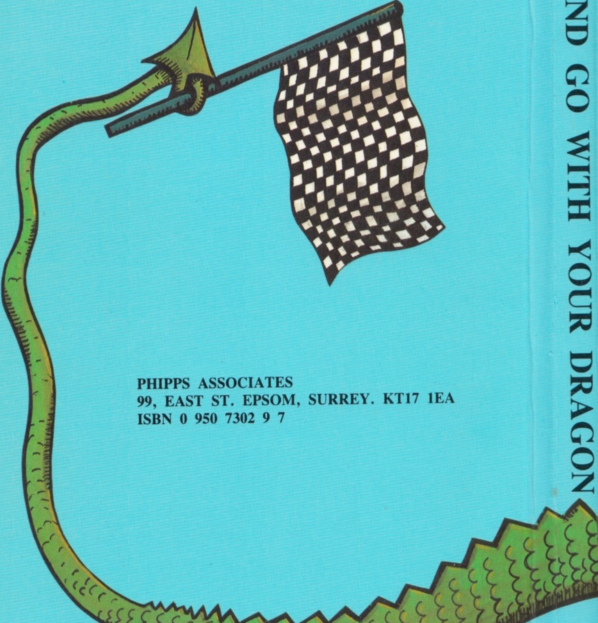
```
600 NEXT Z
              'look to see if anyone still in
610 FOR Y=1 TO H:PL=Y:GOSUB 1190:IF NOT BU THEN 630
620 NEXT Y:PRINT"I STICK":GOTO 680
              'evaluate bank hand and decide to stick/twist
630 GOSUB 1540:IF BU THEN B5=0:BT=0: PRINT"I BUSTED":GOTO 680
640 IF MT>15 THEN PRINT"I STICK ":GOTO 680
650 PRINT "I TWIST ";
660 GOSUB 1020:GOSUB 1100:GOSUB 1160
670 GOTO 630

680 REM SETTLE UP BETS
690 FOR Y=1 TO H: PL=Y:GOSUB 1190   'rules processor see text
700 IF NOT PT AND BP THEN WR=2:GOTO 770
710 IF PT AND BP THEN WR=1:GOTO 770
720 IF BU THEN WR=1:GOTO 770
730 IF C5 AND NOT B5 THEN WR=-2:GOTO 770
740 IF C5 THEN WR=-1:GOTO 770
750 IF PT THEN WR=-2:GOTO 770
760 IF MT>BT THEN WR=-1 ELSE WR=1

770 REM PRINT THE PAYOUT
780 PRINT PL$(Y);":";HT;MT    'name,lowest score, best score
790 IF BU THEN PRINT "BUSTED ";
800 IF C5 THEN PRINT "5 CARDS ";
810 IF PT THEN PRINT "PONTOON ";
         'WR positive means players pays bank
         'WR negative means bank pays players
820 IF WR>0 THEN PRINT PL$(Y);" PAYS ME ";
         ELSE PRINT "I PAY ";PL$(Y);
830 IF ABS(WR)>1 THEN PRINT" TWICE"; ELSE PRINT " ONCE";
840 PRINT ABS(WR)*B(Y)
850 M(3)=M(3)+WR*B(Y)     'adjust bank money
860 M(Y)=M(Y)+WR*B(Y)*-1 'adjust player money
870 NEXT Y:BP=0
880 INPUT"PRESS ENTER TO GO ON";A$
890 GOTO 210
```

```
900 REM SHUFFLE PACK
910 FOR Z=1 TO 52:P(Z)=0:NEXT Z
920 PRINT"PLEASE WAIT A MOMENT - ":PRINT"I AM SHUFFLING"
930 FOR Z=1 TO 52
940 P=RND(52):IF P(P)<>0 THEN 940
950 P(P)=Z:NEXT Z
960 P=1  'TOP OF PACK MARKER
970 RESTORE:FOR Z=1 TO 4:READ ST$(Z):NEXT
980 DATA "CLUBS","DIAMONDS","HEARTS","SPADES"
990 FOR Z=1 TO 4:READ CD$(Z):NEXT
1000 DATA "ACE","JACK","QUEEN","KING"
1010 RETURN

1020 REM DEAL NEXT CARD
1030 FOR DL=1 TO 900:NEXT DL        'vary delay as you wish
1040 W=P(P):ST=1+INT((W-1)/13)
1050 ST$=ST$(ST)
1060 CD=W-(ST-1)*13+1
1070 IF CD>10 THEN CD$=CD$(CD-10) ELSE CD$=STR$(CD)
1080 P=P+1:IF P>52 THEN P=1:PRINT "GOING THROUGH THE PACK AGAIN"
1090 RETURN

1100 REM ADD TO HAND            'increase PG() add to HD() etc
1110 G=PG(PL)+(PL-1)*5+1
1120 HD$(G)=CD$:HS$(G)=ST$
1130 HD(G)=CD:HS(G)=ST
1140 PG(PL)=PG(PL)+1
1150 RETURN

1160 REM SHOW VALUE OF CARD
1170 PRINT TAB(TB);CD$;" OF ";ST$
1180 RETURN
```

```
1190 REM EVALUATE THE HAND
1200 HT=Ø:BU=Ø:AT=Ø:PT=Ø:C5=Ø
1210 FOR Z=(PL-1)*5+1 TO PG(PL)+(PL-1)*5
1220 IF HD(Z)=11 THEN HT=HT+1:AT=AT+1:GOTO 1250
1230 IF HD(Z)>11 THEN HT=HT+10:GOTO 1250
1240 HT=HT+HD(Z)
1250 NEXT Z
1260 IF HT>21 THEN BU=-1
1270 IF PG(PL)=5 AND NOT BU THEN C5=-1
1280 MT=HT:IF AT=Ø THEN 1310
1290 FOR Z=1 TO AT:MT=HT+Z*10:IF MT>21 THEN MT=MT-10:GOTO 1310
1300 NEXT Z
1310 IF MT=21 AND PG(PL)=2 THEN PT=-1
1320 RETURN

1330 REM SR TO ADJUST PLAYERS
1340 IF M(3)>Ø THEN 1380              'bank is OK
1350 PRINT "OH DEAR-YOU HAVE BUSTED THE BANK"
1360 PRINT"I SHALL HAVE TO GET SOME"
1370 PRINT"MORE RAM CHIPS.  SEE YOU LATER":END
1380 SH=Ø:FOR Y=1 TO H
1390 IF M(Y)>Ø THEN 1450
1400 PRINT "POOR OLD ";PL$(Y)
1410 PRINT"IS FLAT BROKE"
1420 R=5:GOSUB 1580       'remarks for losing a player
1430 SH=SH+1:PL$(Y)="":M(Y)=Ø
1440 IF Y=1 AND H=1 THEN PRINT"GAME OVER -":
        PRINT"SUPERIOR INTELLECT WINS":R=4:GOSUB 1580:END
1450 NEXT Y
              'test for two players gone at once
1460 IF SH=H THEN INPUT"WHERE'S EVERYBODY GONE TO";A$:END
1470 IF SH=Ø THEN RETURN 'nobody out
1480 H=H-SH:IF M(2)<1 THEN INPUT
        "PRESS ENTER TO CONTINUE";A$:RETURN
1490 PRINT "MOVE OVER ";PL$(2)      'left hand player gone
1500 PRINT "LOTS OF ROOM ON MY LEFT"
1510 M(1)=M(2):PL$(1)=PL$(2):PL$(2)="":M(2)=Ø
1520 INPUT"PRESS ENTER WHEN READY";A$
1530 RETURN
```

```
1540 REM EVALUATE BANK
1550 PL=3:GOSUB 1190
1560 BT=MT:B5=C5:BP=PT
1570 RETURN

1580 REM RUDE REMARKS
          'always say something for rare events
          'player going or end of game
1590 IF R>3 THEN RR=RND(5):GOTO 1610
1600 RR=RND(25):IF RR>5 THEN RETURN      'say nothing 80% time
1610 RR=(R-1)*5+RR              'remark =event *5 +rr
1620 PRINT A$(RR)
1630 RETURN

1640 REM SET UP REMARKS
               'R=1 for start of new hand
1650 A$(1)="I FEEL LUCKY TODAY"
1660 A$(2)="EYES DOWN"
1670 A$(3)="HERE'S LOOKING AT YOU"
1680 A$(4)="ANYONE FOR TENNIS"
1690 A$(5)="THERE IS NOTHING UP MY SLEEVES"
               'R=2 for placing of bets 9,10 are deliberately
               'not used to make it less talkative
1700 A$(6)="BIG SPENDER"
1710 A$(7)="SURE YOU CAN AFFORD IT"
1720 A$(8)="THAT WILL DO NICELY"
               'R=3 for player busted
1730 A$(11)="OH DEAR I AM SORRY"
1740 A$(12)="IT'S AN ILL WIND"
1750 A$(13)="I DIDN'T SAY ANYTHING"
1760 A$(14)="A PLEASURE TO DO BUSINESS"
1770 A$(15)="A BRIDGE TOO FAR"
               'R=4 for end of game
1780 A$(16)="I GUESS THE BEST CHAP WON"
1790 A$(17)="WHERE ARE THE NEXT SUCKERS"
1800 A$(18)="THANKYOU FOR THE GAME"
1810 A$(19)="COME AGAIN SOON"
1820 A$(20)="YOU CAN'T WIN THEM ALL"
```

```
               'R=5 loss of player gone out
1830 A$(21)="SORRY TO LOSE YOU"
1840 A$(22)="SUCH A NICE CHAP"
1850 A$(23)="BETTER LUCK NEXT TIME"
1860 A$(24)="THAT'S THE WAY IT CRUMBLES"
1870 A$(25)="UNLUCKY AT CARDS-LUCKY IN LOVE"
1880 RETURN
```



PHIPPS ASSOCIATES
99, EAST ST. EPSOM, SURREY. KT17 1EA
ISBN 0 950 7302 9 7

LOAD AND GO WITH YOUR DRAGON

PHIPPS