The Cambridge Collection

30 Programs for the ZX81

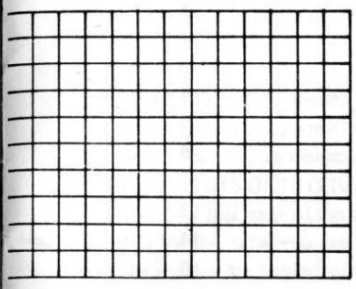
RICHARD FRANCIS

Published by Richard Francis
22 Fox Hollow,
Bar Hill
Cambridge CB3 8EP.

ISBN 0 9507658 1 3

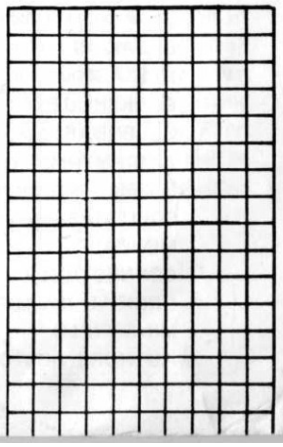
The programs in this book are also available on a quality cassette for automatic loading. Please send cheque or Postal Order for £4.95 (inclusive of post and packing), with your order to the Publisher.

Printed in England by :
Sindall Printing Limited,
347 Cherry Hinton Road, Cambridge CB1 4DJ.
Telephone : (0223) 248091.



The Cambridge Collection

30 Programs for the ZX81



RICHARD FRANCIS

Contents

Introduction

As you have bothered to read this part of the book you are probably expecting some fascinating new insight into computing, so in order not to disappoint you here it is : IT IS POSSIBLE TO DO LOTS OF INTERESTING THINGS WITH ONLY 1K OF RAM. At this stage of the book it is only a hypothesis but I hope that by the time you reach the end you will take it as proven. The motivation behind this book has been the challenge of cramming a programme into 1K (in fact by the time the system variables and machine stack have been subtracted there are less than 900 bytes for program, variables and display). This has involved taking a program idea and stripping it down to its bare essentials whilst maintaining the original purpose, so don't be surprised if you don't see too many remarks in the programs. I owe thanks to computer tradition for some of the program ideas, Lunar Landing and Orbital Invaders are prime examples. Many of the program ideas and all of the programs themselves, however, are entirely original.

In order to discourage readers from the uncritical habit of blindly accepting other peoples programs as perfect (I don't believe there is such a program), I have included Technical Notes with each example to help the enquring mind to understand just what exactly is going on in each part of the program. Whilst I don't expect everyone to read every technical note, I shall be disappointed if nobody writes to tell me that a certain program could be improved by altering certain lines.

Lunar Landing

Lunar Landing programs must be so common by now that their aim needs only the briefest explanation. You, the astronaut, must control the descent of your craft by use of the throttle jets so that you neither crash nor run out of fuel. In this particular version a safe landing is defined as arriving at a point 10 metres or less from the moon's surface with a speed of less than or equal to 10 metres per second.

The display shows the throttle jet setting, the fuel level, the speed and the height. The craft starts falling from 1500 metres above the moon's surface and you must hold down keys 0 to 9 to alter the throttle setting. (They need to be held down as they don't react immediately.) You will soon learn that a throttle setting of 3 causes the speed of descent to drop gradually. Only use the high throttle settings in an emergency or you may find yourself shooting out into space again wasting valuable fuel. A space craft moves from right to left across the screen as an additional graphical indication of position, the extreme left represents the moon's surface.

TECHNICAL NOTES

Lines 10 to 50 initialise the variables and clear the screen.
Lines 60 and 70 print the display information.
Line 80 detects an end game and jumps to 170.
Line 110 draws the spacecraft.
Lines 120 to 150 adjust the throttle, fuel, speed and height depending on the keys pressed.
Lines 170 and 210 deal with the end game.


```
 10 LET V=0
 20 LET S=1500
 30 LET TH=0
 40 LET F=1000
 50 CLS
 60 PRINT "THROTTLE FUEL      SPEED   HEIGHT"
 70 PRINT AT 1,0;TH;TAB 8;F;"  ";TAB 16;V; "     ";TAB
24;S;"     ",,,,
 80 IF S<10 OR F<0 THEN GOTO 170
 90 LET A$=INKEY$
100 PRINT AT 5,(S-V)/100,
110  PRINT AT 5,S/100;"$"
120 IF A$>="0" AND A$<="9" THEN LET TH=CODE
A$-28
130 LET V=V+20*TH-50
140 LET S=S+V
150 LET F=F+10*TH
160 GOTO 70
170 IFV>=-10 AND S<=10 THEN PRINT"SAFE"
180 IF V<-10 THEN PRINT "CRASH"
190 PRINT,,"N/L TO START AGAIN"
200 INPUT A$
210 RUN
```

Mastercode

The computer generates a random number with 4 digits, no digit repeated and you have to guess it. Every time you enter a guess the computer gives you two scores labelled X and Y. For every digit that you have correct the computer gives you a point in the X column and for every instance that you have the correct digit in the wrong column (or position) the computer will give you a point in the Y column. You will see that the scores in both columns added together cannot exceed 4. Some examples of scoring are given.

Number	5	6	7	8	Scores	X	Y
Guess 1	1	6	2	3		1	0
Guess 2	0	1	5	7		0	2
Guess 3	5	6	7	8		4	0

When you have the correct answer press newline to start again.

TECHNICAL NOTES

Lines 30 to 150 A subroutine to evaluate the scores for each guess.
Lines 160 to 240 Generation of a 4 digit (non-repeating) number.
Lines 260 to 300 Main program loop.
Variables: X,Y, Scores.
A$, Guessed number.
B$, Number to be guessed.
M,N, Dummy **FOR-NEXT** variables.


```
10 CLS
20 GOTO 160
30 LET Y=0
40 LET X=0
50 FOR N=1 TO 4
60 IF A$[N]=B$[N] THEN GOTO 130
70 FOR M=1 TO 4
80 IF M<>N AND A$[N]=B$[M] THEN GOTO 110
90 NEXT M
100 GOTO 140
110 LET Y=Y+1
120 GOTO 140
130 LET X=X+1
140 NEXT N
150 RETURN
160 RAND
170 DIM B$[4]
180 DIM A$[4]
190 FOR N=1 TO 4
200 LET B$[N]=STR$[INT[10*RND]]
210 FOR M=1 TO 4
220 IF B$[M]=B$[N] AND N<>M THEN GOTO 200
230 NEXT M
240 NEXT N
250 CLS
260 PRINT "**** X Y"
270 INPUT A$
280 GOSUB 30
290 PRINT A$;" ";X;" ";Y
300 IF X<>4 THEN GOTO 270
310 PRINT "CORRECT"
320 PRINT "N/L TO START AGAIN"
330 INPUT A$
340 RUN
```

Odd Man Out

This is a pattern recognition puzzle for the discerning eye. The computer generates a three-character-position by three-character-position square pattern by randomly combining 9 graphics symbols. This pattern is printed in the top left corner of the screen. The pattern is then jumbled up three times to produce three additional patterns using the same graphics symbols but in different orders. One of these patterns is then corrupted by exchanging one of the graphics symbols (randomly selected) for another randomly selected character. Your problem is to decide, by close scrutiny, which symbol in which pattern is the odd man out. The symbols are numbered as follows:

```
 0  1  2        9 10 11
 3  4  5       12 13 14
 6  7  8       15 16 17

18 19 20       27 28 29
21 22 23       30 31 32
24 25 26       33 34 35
```

Enter the number that you think is the odd man out and the program will stop with report 0/190 if correct, otherwise you get another go. On some rare occasions there won't be an odd man out so beware. If it proves too difficult try it with letters, change lines 100 & 120 to:

100 **IF** A = 1 **AND** B = 1 **THEN LET** B$ (A,B,C,D) = **CHR$** (38 + 25***RND**)

120 **IF** 18*A + 9*B + 3*C + D = G + 31 **THEN LET** B$ (A,B,C,D) = **CHR$** (38 + 25* **RND**)

TECHNICAL NOTES

Line 30 G is the number of the odd man out.

Lines 40 to 170 nested FOR loops generate the patterns printed by line 140.

E & F are the random offsets added to the 3rd and 4th subscripts (modulo 2) of array B$ to give the jumbled patterns.


```
 20 DIM B$[2,2,3,3]
 30 LET G=9+INT[27*RND]
 40 FOR A=1 TO 2
 50 FOR B=1 TO 2
 60 LET E=INT[3*RND]
 70 LET F=INT[3*RND]
 80 FOR C=1 TO 3
 90 FOR D=1 TO 3
100 IF A=1 AND B=1 THEN LET B$ [A,B,C,D]=CHR$ [11
*RND+128 * INT [RND*2]]
110 IF A<>1 OR B<>1 THEN LET B$[A,B,C,D]=B$
[1,1,C+E-3*INT[[C+E]/3.5],D+F-3*INT[[D+F]/3.5]]
120 IF 18*A+9*B+3*C+D-31=G THEN LET
B$[A,B,C,D]=CHR$[11*RND]
130 NEXT D
140 PRINT AT 4*A-4+C,4*B-4;B$[A,B,C]
150 NEXT C
160 NEXT B
170 NEXT A
180 INPUT A
190 IF A<>G THEN GOTO 180
```

Steering

An opportunity to try your hand and nerve at the ambition of your dreams, driving a racing car. All you have to do is steer between the edge markers to avoid a crash.

When starting the game enter a number for the desired width of the road, 8 is a good one to start with. A bigger number will give a wider road and hence make the game easier. You have to be ready to steer as soon as you have pressed newline after entering the roadwidth. Press key 5 to move to the left and 8 to move to the right. See if you can stay the course for a minute without crashing, when you have achieved that, start again with a narrower road. You cannot win this game outright as the road can always be made narrower still but with a little practice you should be able to beat your friends.

TECHNICAL NOTES.

Variables : N, width of track.
A, position of left of track.
B, position of car.
A$, key pressed.

Lines 40 to 150 are the main program loop. Notice that line 120 peeks into the display file area of RAM to see if the position towards which the car is heading is the edge of the race track or not.

Line 50 generates each new position of the track by randomly incrementing or decrementing the old position.


```
 10 INPUT N
 20 LET A=10
 30 LET B=A+N/2
 40 PRINT AT 12,A;"■";TAB[A+N];"■"
 50 LET A=A+INT[3*RND]-1
 60 IF A<0 THEN LET A=0
 70 IF A+N>31 THEN LET A=31-N
 80 LET A$=INKEY$
 90 IF A$=" " THEN GOTO 110
100 LET B=INT[[CODE A$-34]/1.5]+B
110 SCROLL
120 IF CHR$[PEEK[PEEK16396+256*PEEK 16397+1+B]]<>
"■" THEN GOTO 140
130 PRINT AT 0,0;"CRASH"
140 PRINT AT 0,B;"V"
150 GOTO 40
```

Use an inverse video V in line 140.
Ensure that the same graphic symbol (GRAPHIC SHIFT H) is used on both occasions in line 40 and in line 120.

Butterfly

In this game a butterfly flits around the screen in a random fashion, totally unaware that a bird is trying to catch it. You have to control the movement of the bird, by the cursor movement keys (5 to 8) so that the bird catches the right hand wing of the butterfly in its beak. The bird opens and closes its mouth all the time attempting to catch its prey. When it is successful the amount of time taken is displayed and you are invited to press newline to have a go at another butterfly.

TECHNICAL NOTES.

Lines 10 to 50 initialise variables.
Lines 60 to 170 form the main program loop.
Lines 60 & 70 and 110 & 120 draw the bird and butterfly.
Line 130 tests for a successful catch. Subroutines at 180 and 230 calculate the new positions of the butterfly and bird. Notice that subroutine 180 is called twice in the program loop allowing the butterfly to move faster than the bird.
Notice lines 230 & 240, the expressions inside brackets are logical and can be true (equal to 1) or false (zero), hence the J,K co-ordinates of the bird are altered by the cursor movement keys.


```
10 LET C=0
20 LET J=3
30 LET K=15
40 LET X=0
50 LET Y=0
60 PRINT AT Y,X;"▚▞"
70 PRINT AT J,K; "-▄/"
80 GOSUB 180
90 IF INKEY$<>" " THEN GOSUB 230
100 CLS
110 PRINT AT Y,X;"▚▞"
120 PRINT AT J,K;"-▄/"
130 IF INT[X+.5]=K-1 AND INT[Y+.5]=J THEN GOTO 260
140 GOSUB 180
150 CLS
160 LET C=C+1
170 GOTO 60
180 LET X=ABS[X+4*RND-2]
190 LET Y=ABS[Y+2*RND-1]
200 IF ABS X>20 THEN LET X=20
210 IF ABS Y>8 THEN LET Y=8
220 RETURN
230 LET K=K+[INKEY$="8" AND K<20]-[INKEY$="5"
AND K>0]
240 LET J=J+[INKEY$="6" AND J<8]-[INKEY$="7" AND
J>0]
250 RETURN
260 PRINT AT 13,0;C;" SECONDS"
270 INPUT U$
280 RUN
```

Ten Pins

Ten skittles are arranged on the left of the screen and a ball on the right. You have to roll the ball to try to knock down as many skittles as you can. First line up the ball by using cursor movement keys 6 (down) and 7 (up), when you are ready press 0 to let the ball roll. When a skittle is hit it will disappear and the ball may change its course. In each round you have 5 balls numbered 5,4,3,2,1 in inverse video and you have to try to knock down all ten skittles. At the end of each round press newline to start again. This game can be played by two or more people taking turns or by one player just for practice.

TECHNICAL NOTES.

Lines 10 to 30 fix the size of the display file.
Line 49 sets up the skittles.
Lines 90 to 130 form the first program loop that allows the ball to be lined up.
Lines 200 to 260 from the second program loop that rolls the ball. Notice line 220 where each new position is peeked to see if it will hit a skittle and a random offset added if it does.
Lines 270 & 280 decrement the ball counter.

The ten graphic characters in line 40 and the one in line 220 must all be the same (GRAPHICS SHIFT H)
The "5" in line 50 and the "0" in line 280 can both be inverse video characters.

```
 10 FOR N=0 TO 7
 20 PRINT AT N,21;"  "
 30 NEXT N
 40 PRINT AT 0,0;
"■",,"  ■",,"■  ■",,"  ■  ■",,"■  ■",,"  ■",,"■"
 50 LET A$="5"
 60 LET X=0
 70 LET Y=21
 80 PRINT AT X,Y;"  "
 90 IF INKEY$="6" AND X<7 THEN LET X=X+1
100 IF INKEY$="7" AND X>0 THEN LET X=X-1
110 IF INKEY$="0" THEN GOTO 200
120 PRINT AT X,Y;A$
130 GOTO 80
200 FOR Y=21 TO 0 STEP -1
210 LET N=X
220 IF CHR$ [PEEK [1+23 *X+Y+PEEK
16396+256*PEEK16397]]="■" THEN LET
N=ABS[X-1.5+3*RND]
230 PRINT AT X,Y;A$
240 PRINT AT X,Y;"  "
250 LET X=N
260 NEXT Y
270 LET A$=CHR$[CODE A$-1]
280 IF A$<>"0" THEN GOTO 60
290 INPUT A$
300 RUN
```

Fortress

You are a medieval lord defending your fortress when an enemy (marked "*" on the screen) creeps up to the base of your look out tower. Unknown to the invader, he is standing next to your secret protection mechanism, which when fired can fling the invader back into the moat. You must drop a large weight over the side of the tower to activate the mechanism. Choose the weight of your choice by pressing keys 1 to 9 then press 0 to release it. If you choose the wrong weight then you will repel the enemy but he won't end up in the moat. Press newline after each go to move onto another tower whose distance from the moat will probably be different from the last one.

Keep a score of your success rate in drowning the enemy.

TECHNICAL NOTES.

Line 40 and the subroutine at 200 draw the out look tower, protection mechanism and moat.
Lines 60 to 90 allow the appropriate weight to be selected.
Lines 100 to 140 allow the weight to drop.
Lines 150 to 180 roll the invader backward and possibly into the moat.


```
10 LET B$="0"
20 CLS
30 LET A$=" "
40 PRINT,,"■  ■",,"■■■",,"■■",,"■"
50 GOSUB 210
60 IF INKEY$<>"" THEN LET A$=INKEY$
70 IF A$>="1" AND A$<="9" THEN LET B$=CHR$[CODE
A$+128]
80 PRINT AT 0,2;B$
90 IF A$<>"0" THEN GOTO 60
100 PRINT AT 0,2;"  ";B$
120 FOR B=0 TO 9
130 PRINT AT B,3;"  "; AT B+1,3;B$
140 NEXT B
150 FOR B=6 TO INT[RND*2]+CODE B$-148
160 PRINT AT 11,B;"  "; TAB [B+1];"*"
170 NEXT B
180 IF B=A THEN PRINT AT 11,A;"  ";AT 12,A;"*"
190 INPUT A$
200 RUN
210 PRINT AT 10,3;"▣▣▣"; AT 11,5;"▣*"
220 PRINT AT 12,7;"■■■■■■■■■■■■■■■"
230 LET A=11+INT[9*RND]
240 PRINT AT 12,A;"  "
250 RETURN
```

N. B. The string in line 220 consists of 15 GRAPHIC SHIFT/H characters.

The "0" in line 10 should be inverse video.

Balloon

This is a game requiring skill and accurate judgement and needs to be played a few times before you can hope to achieve a safe landing. You are controlling the flight of a hot air balloon which you must raise into the air and land again at a specified destination. The screen displays five flight parameters, the mass is the number of sandbags that you have on board, these can be discarded one at a time by holding down the "M" key. The "Heat" is the burner setting and controls the upward thrust, this can be set from 0 - 9 by holding down the appropriate key. Whenever the heat is turned on fuel is used up, and the remaining fuel is displayed. The altitude and distance from the specified destination are also shown. As you rise you will catch faster breezes and your speed will increase.

A safe landing is defined as hitting the ground within 20 metres of the destination (the distance is displayed in units of ten metres). Common errors are trying to carry too many sandbags and not being able to get high enough (or even off the ground) without running out of fuel or alternatively travelling too high and overshooting the target. A balloon travels from right to left across the screen as an added indication of position, the left edge representing the target. The keys don't react immediately as they are scanned every second (approx.) so hold them down until the screen registers a reaction. After landing press any key to play again.

TECHNICAL NOTES

Lines 10 to 70 initialise the screen and variables.
Lines 80 to 190 form the main program loop, continuously monitoring the keyboard and updating flight parameters.
Lines 200 to 250 analyse the landing.


```
10 CLS
20 LET D=1500
30 LET M=10
40 LET F=300
50 LET A=0
60 LET H=0
70 PRINT
"MASS      HEAT      FUEL      ALT      DIST"
80 PRINT AT 1,0;M;"  "; TAB 7 ;H; TAB 14 ; F;"     "; AT
1,21;A;"     "; AT 1,28;D;"     "
90 IF A=0 AND D<1500 THEN GOTO 200
100 LET A$=INKEY$
110 IF A$="M" THEN LET M=M-1
120 IF M<=0 THEN LET M=0
130 IF A$>="0" AND A$<="9" THEN LET H=VAL A$
140 LET F=F-H
150 LET A=INT[A+20*H/[10+M]-5-M/5]
160 IF A<=0 THEN LET A=0
170 LET D=D-A
180 PRINT AT 3,ABS[D/50];"B "
190 GOTO 80
200 IF ABS D>=2 THEN GOTO 240
210 PRINT "SAFE LANDING"
220 PAUSE 9999
230 RUN
240 PRINT "CRASH"
250 GOTO 220
```

N.B. The character "B" in line 180 and the text in lines 210 and 240 will benefit from being inverse video.

Spell

This simple little game will provide both an amusing competition to play against opponents and also a fascinating insight into the way the human mind remembers things. The program generates a string of up to 15 letters selected at random, the string length being a random variable also. The string (or word) is displayed for a few moments, longer for longer words, and when it disappears you have to enter it into the computer as best you can remember it. Your answer is checked by the computer and a score is given. Press newline for another go. It will be instructive to keep score noting what lengths of words you can most easily remember. The equation in line 80 determines the length of time that each word is displayed for, the variable A is the word length. Try altering this equation until you get a consistent pass/fail rate independent of word length.

TECHNICAL NOTES

Variables A, length of string. (How long is it . . .)
A$, String.
B$, Your guess.
N, Dummy **FOR NEXT** variable

Lines 10 to 50 generate the string
Lines 70 to 90 print it for a calculated amount of time.
Lines 110 to 160 analyse results.


```
10 LET A=INT[15*RND+1]
20 LET A$=" "
30 FOR N=1 TO A
40 LET A$=A$+CHR$[38+25*RND]
50 NEXT N
60 CLS
70 PRINT A$
80 FOR N=1 TO 2*A+0.5*A**3
90 NEXT N
100 CLS
110 INPUT B$
120 IF A$<>B$ THEN PRINT B$, "WRONG"
130 PRINT A$,"CORRECT"
140 INPUT B$
150 CLS
160 GOTO 10
```

Use inverse video text in lines 120 and 130.

Truth Table

This is a real brain teasing puzzle that tests your powers of logical analysis. The computer generates a logical expression equating Z to a function of P, Q&R and you have to work out the truth table for that function.

A typical equation might be :

Z=P **AND** (Q **AND** R **OR** P **AND** Q) **OR** Q

in which case the truth table would be :

P	Q	R	Z
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

Boolean algebraic expressions such as the one above are often written using the following notation :

A.B means A AND B

A+B means A OR B

$\overline{A}$ means NOT A

In analysing such expressions the following theorems are useful :

1 A+B.C = (A+B).(A+C)

2 A.(B+C) = A.B + A.C

3 A + A.B = A

4 A + $\overline{A}$.B = A + B

5 A.B + A.$\overline{B}$ = A

6 A.B + $\overline{A}$.C = (A + C).($\overline{A}$ + B)

7 A.B + B.C + $\overline{A}$.C = A.B + $\overline{A}$.C

Considering the example expression above we can see that P. (Q.R+P.Q) + Q can be reduced to P.Q.(R + P) + Q using theorem 2, and again to simply Q using theorem 3.

When you have written down your guess at the truth table on paper, press newline and the computer will give its version for you to check against.

TECHINCAL NOTES

Lines 10 to 80 generate the random logical expression as a string contained in B$, this is augmented by bracketed subexpressions formed by the routine at lines 200 to 379.

Lines 90 to 190 print the truth tables, evaluating B$ by use of the VAL function.

```
 10 RAND
 20 LET B$="P"
 30 FOR N=1 TO 5
 40 LET B$=B$+CHR$[217+RND]
 50 IF RND<.3 THEN LET B$=B$+CHR$215
 60 IF RND<.6 THEN GOTO 200
 70 LET B$=B$+CHR$[53+2*RND]
 80 NEXT N
 90 PRINT "Z= ";B$
100 PRINT "RQP Z"
110 INPUT A$
120 FOR R=0 TO 1
130 FOR Q=0 TO 1
140 FOR P=0 TO 1
150 PRINT R;Q;P;" ";VAL B$
160 NEXT P
170 NEXT Q
180 NEXT R
190 STOP
200 LET B$=B$+"["
210 LET B$=B$+CHR$[53+2*RND]
220 FOR M=N TO 5* RND
230 LET B$=B$+CHR$[217+RND]
240 IF RND<.3 THEN LET B$=B$+CHR$215
250 LET B$=B$+CHR$[53+2*RND]
260 NEXT M
270 LET B$=B$+"]"
280 LET N=M
370 GOTO 80
```

Momentum

This program (and SPHERES) will test your skill at mental arithmetic. You may choose a level of difficluty from 1 to 3 when prompted. The computer draws two trains, one stationary the other approaching it from the left. When the two collide the momentum of the first is transferred entirely to the second without any loss. Given the speed and mass of the first train and the mass of the second, you must calculate the resultant speed of the second train. Your answer is marked and you are given the chance to have another go by pressing newline. Keep a score of your results and use this program to improve your mental arithmetic.

TECHNICAL NOTES

Lines 50 to 100 generate the random masses and speeds. Notice how the level of difficulty, A, is included in the equations in Lines 50, 60 and 70.

Lines 130 to 200 draw the trains in motion.

Lines 220 and 240 analyse your answer and mark it correct if it is within 0.01 of the correct value.


```
 10 CLS
 20 RAND
 30 PRINT "LEVEL 1-3"
 40 INPUT A
 50 LET M1=INT[RND*10**A]
 60 LET M2=INT[RND*10**A]
 70 LET V1=INT[RND*10**A]
 80 LET V2=M1*V1/M2
 90 PRINT "1ST
TRAIN ";M1;" TONS  AT ";V1;" MPH"
 100 PRINT "2ND TRAIN ";M2;" TONS"
 110 PRINT "PRESS N/L"
 120 INPUT A$
 130 FOR N=0 TO 15
 140 LET A=SIN A
 150 PRINT AT 12,N; " ▗▄▖";AT12,19;"▗▄▖";AT
13,N;" ▄▄▄";AT13,19;"▄▄▄"
 160 NEXT N
 170 FOR N=19 TO 28
 180 LET A=SIN A
 190 PRINT AT12,N; " ▗▄▖";AT13,N;" ▄▄▄"
 200 NEXT N
 210 PRINT "ENTER SPEED OF 2ND TRAIN"
 220 INPUT A
 230 IF ABS[A-V2]>=0.01 THEN PRINT A;" WRONG"
 240 PRINT V2; "CORRECT"
 250 INPUT A$
 260 RUN
```

Letters

Have you ever wanted to tell someone something but not known quite how to put it? Well here's your chance - - - say it with letters - - - and with letters this big they are bound to get the point.

This program allows you to enter a sentence, or any string of characters, and displays it in characters that are either four or eight times as high as usual. First enter the letter L (large) or S (small) to select the size. Next enter your message, using up to 50 characters with codes in the range 0 to 63. Refer to pages 181/182 of the manual if in doubt, but in short this includes all the letters, numbers and punctuation marks, but not inverse video characters or keywords. The message will then scroll up the left hand side of the screen.

This program must have a lot of practical joke potential, imagine someone coming home, switching on the T.V. (unaware that a ZX81 with this program running is hidden somewhere behind it) and seeing a personal message on the screen in two inch high letters.

TECHNICAL NOTES

This program relies on the ZX81 character set being stored in ROM starting at location 7,680. Eight bytes are used to represent a character so that for example the letter "P" is stored something like this :

```
00000000
01111100
01000010
01000010
01111100
01000000
01000000
00000000
```

Hold this at a distance and you should be able to make out the letter.

Notice the FOR NEXT loop from lines 50 to 100 where each bit from left to right of a byte is analysed. The larger loop from 20 to 110 prints a whole character.

Line 40 is of particular interest as it scrolls up on every line for large characters or alternate lines for small letters. Notice also the second co-ordinate in the PLOT statement of line 90. This is a logical expression, and so takes the value 1 or 0, alternately as C is odd or even.

Lines 130 to 220 are the main program that accepts the message.

Lines 20 to 120 form the powerful subroutine that prints out a character.

```
10 GOTO 130
20 FOR C=A TO A+7
30 LET B=PEEK C
40 IF B$="L" OR B$="S" AND C/2=INT[C/2] THEN
SCROLL
50 FOR N=1 TO 8
60 LET B=2* [B-256*INT[B/256]]
70 IF B>=256 AND B$="L" THEN PRINT "■";
80 IF B<256 AND B$="L" THEN PRINT "▨";
90 IF B>=256 AND B$="S" THEN PLOT
N-1,C/2=INT[C/2]
100 NEXT N
110 NEXT C
120 RETURN
130 PRINT "L OR S"
140 INPUT B$
150 CLS
160 PRINT "MESSAGE"
170 INPUT A$
180 FOR M=1 TO LEN A$
190 LET A=7680 + 8*CODE A$[M]
200 GOSUB 20
210 NEXT M
220 GOTO 180
```

Spheres

Like the momentum program this one tests your powers of arithmetic. The computer draws a sphere and gives you its diameter from which you have to calculate its volume. Your answer is marked as correct if it is within 0.5 of the computers' answer. At the beginning you can select your level of difficulty and at the end you can press newline to be set another problem.

TECHNICAL NOTES

Lines 10 to 105 draw the sphere and give its' diameter. Notice the use of **SIN** and **COS** in the coordinates of the **PLOT** statement in line 80.
Lines 150 to 210 analyse your results.


```
10 RAND
20 CLS
30 PRINT "ENTER LEVEL 1 OR 2"
40 INPUT A
50 SCROLL
60 PRINT AT 0,0;"LEVEL ";A
70 FOR N=0 TO 20
80 PLOT 5+4*SIN[N*PI/10],10+4*COS[N*PI/10]
90 NEXT N
100 PRINT AT 21,0;"<---> DIAMETER IS ";
110 LET B=INT[RND*10**A]
120 PRINT B
130 SCROLL
140 PRINT "ENTER VOLUME"
150 INPUT C
160 SCROLL
170 SCROLL
180 IF ABS[C-PI*B**3/6]>=.5 THEN PRINT C,"WRONG"
190 SCROLL
200 SCROLL
210 PRINT INT[PI*B**3/6+.5],"CORRECT"
220 INPUT A$
230 RUN
```

Monitor

This program allows you to look at parts of memory and poke bytes into memory. This will be very useful if you wish to look at the ROM routines, or write your own machine code programs and poke them into RAM. When you run the program you will first be invited to enter an address in decimal. The computer will then print out this address in hexadecimal followed by its contents (also in hex) and the character representation of its contents. The screen will then start scrolling up displaying each successive byte in the memory. You can stop this process by holding down the newline key. If you press the "X" key you can enter another decimal address and examine a different area of the memory. Pressing the "I" key will allow you to change the last byte by entering a new value in hex. Pressing "R" followed by entering the letters "R-U-N" will execute a machine code subroutine starting at the last address displayed on the screen.

TECHNICAL NOTES

Lines 30 to 100 form the routine that displays the hex value of A (lines 40 to 80) the hex value of its contents (lines 90 to 100) and also the character representation of those contents. The **FOR NEXT** loop in lines 50 to 80 is not the neatest way to decode a decimal number to a 4 digit hex one, but it is certainly one of the quickest, avoiding lengthy operations such as exponentiating.
Lines 220 to 280 check to see if any keys have been pressed.
Lines 300 to 320 enter a byte into memory.
Lines 400 to 420 execute a **USR** subroutine.


```
 10 REM XXXXXXX==64 times==XXXXXXX
 20 GOTO 200
 30 PRINT AT 6,0
 40 LET B=65536
 50 FOR M=1 TO 4
 60 PRINT CHR$[28+INT[16*[A-B*INT[A/B]]/B]];
 70 LET B=B/16
 80 NEXT M
 90 LET B=PEEK A
100 PRINT TAB 6;
CHR$[28+INT[B/16]];CHR$[28+B-16*INT[B/16]];TAB 9
;CHR$ B
110 SCROLL
120 GOTO 220
200 INPUT A
210 GOTO 30
220 LET A$=INKEY$
230 LET A=A+[A$=" "]
240 IF A$=" " THEN GOTO 210
250 IF A$="X" THEN GOTO 200
260 IF A$="I" THEN GOTO 300
270 IF A$="R" THEN GOTO 400
280 GOTO 220
300 INPUT A$
310 POKE A,16*[CODE A$[1]-28]+CODE A$[2]-28
320 GOTO 210
400 INPUT A$
410 IF A$="RUN" THEN PRINT USR A
420 GOTO 210
```

Chase

This is a mad panic chase to catch an alien who is emitting hazardous radiation. Many people have died already and it is now up to you to stop him in his tracks. Fortunately you have your radiation level (Geiger-Muller) detector with you so you can see when you are getting close. You start from the top left corner and the inverse video character (0 to 9 followed by A to Z) indicates your distance, Z meaning a long way off. You can move up, down left and right by pressing the cursor movement keys. When you catch him you are told how many moves you took and you are invited to press newline to have another go.

TECHNICAL NOTES

Lines 10 to 70 initialise the variables and clear the screen.
Lines 70 to 150 form the main program loop.
X,Y are your coordinates, whilst A,B are those of the alien. Notice how lines 110 to 140 alter these coordinates by using logical expressions such as (**INKEY$** ="8").
The **PRINT AT** statement in line 50 has built in safeguards to ensure that it doesn't try to print off the screen.


```
 10 RAND
 20 LET A=INT[20*RND]
 30 LET B=INT[30*RND]
 40 LET X=0
 50 LET N=0
 60 LET Y=0
 70 CLS
 80 PRINT AT ABS[Y-20*INT[Y/20]],
ABS[X-31*INT[X/31]];
CHR$[156+ABS[Y-A]+ABS[B-X]]
 90 LET N=N+1
100 IF A=Y AND X=B THEN GOTO 160
110 LET X=X+[INKEY$="8"]-[INKEY$="5"]
120 LET Y=Y+[INKEY$="6"]-[INKEY$="7"]
130 LET A=A+[RND<.2]-[RND<.2]
140 LET B=B+[RND<.2]-[RND<.2]
150 GOTO 70
160 PRINT AT 0,0;"CAUGHT IN ";N;" MOVES"
170 INPUT I$
180 RUN
```

Vols

This is a useful conversion program that will tell you how many glasses of beer you can drink after we go metric, and still drive home. It converts from imperial measure, Gallons, Pints and Fluid Ounces to metric litres. First enter IMP or MET and then the volume that you want converted. The program will immediately give you the equivalents. Press newline for another go.

TECHNICAL NOTES

Conversion from Imperial to Metric can be done in one line, line 200, but the reverse takes 4 lines (lines 90 to 120). Having done the calculations line 220 prints the results.


```
10 PRINT "ENTER IMP OR MET"
20 INPUT I$
30 IF I$="IMP" THEN GOTO 140
40 PRINT,,"ENTER VOL"
50 INPUT M
60 PRINT "ENTER UNIT LTR OR ML"
70 INPUT I$
80 IF I$="ML" THEN LET M=M/1000
90 LET F=M*160/4.545960
100 LET G=INT[F/160]
110 LET P=INT[F/20]-8*G
120 LET F=F-20*P-160*G
130 GOTO 210
140 PRINT,,"ENTER GALS"
150 INPUT G
160 PRINT,,"ENTER PINTS"
170 INPUT P
180 PRINT,,"ENTER FL OZS"
190 INPUT F
200 LET M=4.54960*[G+P/8+F/160]
210 CLS
220 PRINT M,"LITRES",,,"          EQUALS",,G,"
GALLONS AND",,,P,"PINTS AND",,,F,"FLUID OUNCES"
230 INPUT I$
240 CLS
250 RUN
```

Temp

This program converts between the three temperature scales currently in use, Centigrade (or Celsius), Fahrenheit and Kelvins. Zero degrees centigrade or thirty two fahrenheit is the melting point of ice, whilst zero Kelvins is absolute zero, the lowest temperature that any material can reach. That which appears on the macroscopic scale as the property we all understand as temperature, appears on the sub-microscopic scale as the rate of movement (oscillation) or excitation of molecular particles. Absolute zero is the temperature at which they all come to rest - - - so now you see why it is "absolute".

TECHNICAL NOTES

The key to this program is line 50 where the temp in degrees Fahrenheit is evaluted from the input data. This is then converted to the two other scales (lines 60 and 70) and printed out (lines 80 and 90).

This program could probably just be written in three lines, by entering the temp T following by the units A$ and combining lines 50 to 90 into one enormous print statement.

You may wish to try it as an exercise.


```
10 PRINT AT 0,0;"ENTER TEMP"
20 INPUT T
30 PRINT T,,,,"ENTER F,C OR K"
40 INPUT A$
50 LET F=[A$="F"]*T + [A$="C"]*[32+9*T/5]
+[A$="K"]*[32+9/5*[T-273]]
60 LET C=5/9*[F-32]
70 LET K=C+273
80 PRINT,,F; TAB 13;"FAHRENHEIT"
90 PRINT,,C; TAB 13;"CENTIGRADE",,,K; TAB
13;"KELVIN"
```

Passengers

In this game, you are a train driver taking an early morning commuter train into a major city, stopping at five stations along the route. There are a number of trains at about this hour in the morning and so if the passengers on the platform can't see a place to sit, they usually wait for the next train. The cumulative effect of this behaviour is that the number of passengers that get on at each station is directly proportional to the number of carriages that you can line up against the platform. So if you stop short or overshoot then you lose passengers and create more congestion for later commuters. In total if you can pick up 400 passengers by the time you reach the city you will have been successful in preventing the build up of congestion later. You can control the acceleration by pressing keys 5 to 9 and the braking by keys 1 to 4. 1 is maximum braking and 9 is maximum acceleration. The display shows the number of the station that you are heading towards, the number of passengers on board, your acceleration (braking is negative acceleration) your velocity and the distance from the next station. If you stop more than 100 metres from the station then you won't pick anyone up whereas if you stop in exactly the right place you could take on a hundred passengers.

TECHNICAL NOTES

Lines 10 to 40 initialise the variables and the display.
P = no. of passengers, N = no. of next station, A =acceleration, V = velocity, and D = distance from next station.
For each station you continue around the loop 80 to 160 until the train stops.
Line 170 adds to the number of passengers according to the distance from the station.


```
10 LET P=0
20 LET A=0
30 LET V=0
40 PRINT "STAT  PASS  ACEL  VEL  DIST"
50 FOR N=1 TO 5
60 LET D1=1000*INT(RND*5+1)
70 LET D=D1
80 LET A$=INKEY$
90 IF A$=" " THEN GOTO 120
100 LET A=CODE A$-33
110 IF A$="0" THEN LET A=0
120 LET V=V+A*400/(P+400)
130 IF V<=0 THEN LET V=0
140 LET D=D-V*3
150 PRINT AT 1,0;N;TAB 6;P;TAB 12;A;"    "; AT
1,18;INT V;"    ";AT 1,24; INT D;"    "
160 IF V<>0 OR D>=D1 THEN GOTO 80
170 LET P=P+50-5*ABS(D/10) +ABS(50-5*ABS(D/10))
180 NEXT N
190 IF P>=400 THEN PRINT "SUCCESSFUL TRIP"
200 INPUT A$
210 RUN
```

Snap

This game tests your skill at recognising sequences of letters quickly. You have to enter a word of between 5 and 15 characters in length. It is better to enter a well known word or one that you will easily recognise, but it will be cheating to enter a word with letters repeated. The letters of this word will then flash onto the screen in a jumbled up fashion for less than a second. This will happen twenty times at random intervals. During each display you have to match the word on the screen with the one originally entered and estimate how many letters correspond. If you think it is a lot then press the "0" key quickly before the word disappears again. Once you have pressed "0" then no more words will be displayed, but the word that you have chosen will be compared with the original and the number of letters corresponding will be given as your score. If you have more than 50% of the letters correct then you are considered a winner. Press newline for another go.

TECHNICAL NOTES

Lines 70 to 230 form the main program loop in which the inner loop 100 to 170 generates and prints, a character at a time, a jumbled up version of the word entered. A$ contains the original word, and the letters are taken from this at random and put into the string B$. In order not to take the same letter twice, as each letter is taken it is replaced by an inverse video letter in A$. When the next jumped up word is formed they are turned back from inverse video to normal letters. This process alternates.
Line 220 switches C from +1 to - 1 alternately.
Lines 240 to 310 analyse your results.


```
10 FAST
20 CLS
30 INPUT A$
40 LET B=LEN A$
50 IF B<5 OR B>15 THEN GOTO 30
60 LET C=1
70 FOR M=1 TO 20
80 LET B$=" "
90 PRINT AT 0,0;
100 FOR N=1 TO B
110 LET A=INT[RND*B+1]
120 LET D=CODE A$[A]
130 IF C*D>C*160 THEN GOTO 110
140 LET A$[A]=CHR$[CODE A$[A]+C*128]
150 LET B$=B$+A$[A]
160 PRINT A$[A];
170 NEXT N
180 PRINT,M
190 PAUSE 20+B
200 POKE 16437, 255
210 IF INKEY$="0" THEN GOTO 240
220 LET C=C-2*SGN C
230 NEXT M
240 LET M=0
250 SLOW
260 FOR N=1 TO B
270 IF A$[N]=B$[N] THEN LET M=M+1
280 NEXT N
290 PRINT A$,M;" OUT OF ";B
300 PRINT 100*M/B;" PER-CENT"
310 IF M/B>.5 THEN PRINT "WINNER"
320 INPUT A$
330 RUN
```

Break In

This is a poor mans version of a popular game of a similar name (poor in the sense that he only has 1K to play with). A wall of bricks, precariously placed, has to be destroyed by bouncing a ball against it. As each brick is hit it disappears out of the wall, and the ball rebounds. The ball has 10 lives and one is lost each time it hits the bottom edge of the court. To stop such collisions, a paddle can be moved from left to right across the screen, to hit the ball. Use the cursor movement keys 5 & 8 to control it.

TECHNICAL NOTES

Lines 10 to 50 draw the board.
Lines 60 to 110 initialise the variables.
A & B are the ball co-ordinates, X is the paddle position. A1 & B1 represent the direction of motion of the ball.
Lines 110 and 120 PEEK the next ball position to see if it is going to hit anything, if it does, then its vertical direction is reversed (line 160).


```
10 FOR A=1 TO 3
20 FOR B=0 TO 14 STEP 2
30 PRINT AT B/2,16;" ";AT A,B+[A/2=INT[A/2]];"■";
40 NEXT B
50 NEXT A
60 LET L=165
70 LET X=8
80 LET B=9
90 LETA1=-1
100 LET B1=1
110 LET D=PEEK 16396+256*PEEK 16397
120 LET D=PEEK[D+B+18*A+1]
130 LET L=L-[A+D=7]
140 PRINT AT A,B;CHR$ L;AT 7,X;" ■■ "
150 LET X=X-[INKEY$="5"]+[INKEY$="8"]
160 LET A1=A1-2*A1*[D=128]+2*[A=0]-2*[A=7]
170 LET B1=B1+2*[B=0]-2*[B=16]
180 PRINT AT A,B;" "
190 LET A=A+A1
200 LET B=B+B1
210 IF L<>155 THEN GOTO 110
```

Message

This program and the next rely upon a couple of machine code subroutines which must first be poked into a **REM** statement before the rest of the program can be written. First enter the program (line 10 to 70) at the top of the page, run the program and enter the 71 bytes of machine code that are listed. Next delete lines 20 to 70 and make a recording of the **REM** program with just the **REM** statement as this will be useful for the next program. Now continue entering lines 20 to 160 from the program at the bottom of the page.

When run, the program invites you to enter a message of up to 70 characters, (with codes in the range 0 to 63) but not inverse video characters or keywords. After pressing newline, the message will move across the top of the screen from right to left in very large letters. At the end your message will be repeated so don't forget to leave some space at the end of the message.

You may wish to alter the foreground or background shades of the characters. 16579 is the foreground address and 16536 is for the background. **POKE** 16536,136 for example, will cause a grey background to be generated.

TECHNICAL NOTES

As with the letters program earlier in the book, this program relies upon the character set being stored in ROM. For each character to be displayed the Basic program pokes eight bytes from the character set into an area of RAM in the **REM** statement (lines 70 to 110). Most of the rest of the work is done by the two machine code subroutines.

The one at 16558 displays a column of eight character positions at the extreme right of the screen and the routine at 16514 moves the top lines of the display one position to the left.

Lines 20 to 40 are used to pad out the display file.


```
10 REM XXXXXX--75 times--XXXXX
20 FOR N=16514 TO 16584
30 INPUT I$
40 SCROLL
50 PRINT I$
60 POKE N,16*[CODE I$[1]-28]+CODE I$[2]-28
70 NEXT N
```

Insert the following bytes of hexadecimal:

```
00 2A 0C 40 ED 5B 0C 40 3E 08 32 82 40 13 23
23 01 1F 00 ED B0 3E 00 12 13 3A 82 40 3D 20
EB 32 82 40 C9 80 80 80 80 80 80 80 80 80 11
A6 40 2A 0C 40 06 08 0E 20 23 0D 20 FC 1A CB
07 12 30 02 36 80 23 13 10 EE C9
```

Delete lines 20 to 70, do NOT touch line 10.

```
10 REM machine code routines ----
20 FOR N=0 TO 7
30 PRINT AT N,31;" "
40 NEXT N
50 INPUT I$
60 FOR N=1 TO LEN I$
70 LET A=7680+8*CODE I$[N]
80 FOR C=0 TO 7
90 POKE 16550+C,PEEK[A+C]
100 NEXT C
110 FOR C=0 TO 7
120 LET L= USR 16558
130 LET L= USR 16514
140 NEXT C
150 NEXT N
160 GOTO 60
```

Duckshoot

This program uses the machine code subroutines stored in the REM statement of the previous program (Message), so if you havn't already done so, turn back and enter the machine code before you try to write the rest of the program. You will also need to poke eight bytes of data into the REM statement as detailed at the bottom of the program. When the program is run, a continous row of ducks move across the top of the screen. Unlike common fairground ducks these are bionic (just look at the size of them) and they can only be knocked out effectively by being struck on the beak. Your missiles start at the bottom left of the screen and you must judge your moment to press "0" to fire the missile towards the sitting targets. If you miss you automatically get another go, if you hit, then the screen will clear.

If you wish to stop the program then press "Q" rather than BREAK as the latter will mean that you have to re-poke the data into RAM.

TECHNICAL NOTES

Lines 20 to 50 fix the size of the display file. Line 80 pokes different characters into the foreground display address, as the ducks move across the screen.
The duck is made up as follows :-

```
                C =        01234567
POKE 16550,0    ........   00000000
POKE 16551,0    ........   00000000
POKE 16552,98   ........   0■■000/0
POKE 16553,252  ........   =■■■■■00
POKE 16554,60   ........   00■■■■00
POKE 16555,24   ........   000■■000
POKE 16556,0    ........   00000000
POKE 16557,0    ........   00000000
```



```
10 REM machine code routines ----
20 CLS
30 FOR N=0 TO 7
40 PRINT AT N,31;" "
50 NEXT N
60 LET N=21
70 FOR C=0 TO 7
80 POKE 16579,128-108*[C=0]-104*[C=6]
90 PRINT AT N,0;"0"
100 LET L= USR 16558
110 LET L= USR 16514
120 PRINT AT N,0;" "
130 LET N=N-[N<21]+22*[N=0]
140 LET N=N-[INKEY$="0"]
150 NEXT C
160 IF INKEY$="Q" THEN STOP
170 IF N=3 THEN RUN
180 GOTO 70
```

The following data should also be poked into RAM.

```
POKE 16550,0
POKE 16551,0
POKE 16552,98
POKE 16553,252
POKE 16554,60
POKE 16555,24
POKE 16556,0
POKE 16557,0
```

Calendar

In the mid 1st century B.C. Julius Caesar invited an Alexandrian astronomer, Sosigenes to advise him about reforming the Lunar calendar. Sosigenes suggested that the old system should be abandoned, and a new "Julian Calendar" was started based around a 365¼ day year. This was achieved by adding an extra day to February in every fourth year and calling it a leap year. By the mid 16th century, however, the vernal equinox had moved by ten days from its proper position because the Julian basis of 365¼ days was 11 mins and 14 seconds longer than the true tropical year. Pope Gregory XIII was elected in 1572 and sought the advice of a jesuit astronomer Christopher Clavius to correct the errors. This resulted in losing ten days in October 1572 (the15th followed immediately after the 4th) and a new system, the "Gregorian" calendar, in which 3 out of 4 centenery years are not leap years.

This program uses the Gregorian calendar to display the days of any month since October 1572. Simply enter the year and the month as pompted and the computer will display a calendar. Unfortunately (I will blame lack of RAM) every month is shown as having 31 days, so don't get caught out.

TECHNICAL NOTES

D$ contains the first 3 letters of the days of the week. M$ contains 12 numbers, being the accumulated errors, modulo 7, for each month, that arise by assuming that each month has 28 days. For example, if you assume that January has 28 days (it has 31) then when you reach February you will be 3 days out of step, so the 2nd entry is 3.

Line 70 adds extra leap year days.
In line 150 the complex expression starting (MO< 3)* determines if the year is a leap year and if the month is Jan. or Feb. .
Line 170 prints out the days of the week.

```
 10 PRINT "YEAR"
 20 INPUT Y
 30 PRINT "MONTH 1-12"
 40 LET M$="033614625035"
 50 LET D$="SUNMONTUEWEDTHUFRISAT"
 60 LET N=Y
 70 LET M=N+INT[N/4]-INT[N/100]+INT[N/400]
 80 INPUT MO
 90 CLS
100 FOR A=0 TO 6
110 FOR B=0 TO 4
120 LET C=7*B+A+1
130 IF C<=31 THEN PRINT AT A,B*3;C
140 NEXT B
150 LET C=M+A+CODE M$[MO]-28-[MO<3]*
[[N/4=INT[N/4]]-[N/100=INT[N/100]]+
[N/400=INT[N/400]]]
160 LET C=C-7*INT[C/7]
170 PRINT AT A,15;D$[3*C+1 TO 3*C+3]
180 NEXT A
190 PRINT,,MO;"-";Y
```

Biorhythm

According to biorhythm theory, our intellect, emotions and vitality rise and fall in cycles of 33, 28, and 23 days respectively. These cycles start at birth and continue until the day we die. It is interesting therefore to find out which day of each cycle we are in at any time. It is also interesting to do this for famous people such as politicians on election day and so on.

The program starts by asking you to enter 3 numbers (pressing newline after each) for the day month and year of birth (use e.g. 1947 not just 47 for year). The computer will then prove how smart it is by telling you which day of the week that was, and ask you for today's date (or the date of the analysis). The computer will then print out the day of each cycle in the form INTELLECT 17 meaning day 17 of the 33 day intellectual cycle.

TECHNICAL NOTES

This program only just fits into 1K, so don't be tempted to add your own frills, and if it won't work this may be due to having a non-empty gosub stack. This may be cleared by executing **RETURN** as a command until the error 7/0 appears, showing it is empty.

The key to this program is the subroutine at 110 which accepts the Day (1 - 31), Month (1 - 12) and Year, prints out the day of the week and returns in N the number of days since 1st Jan. 0000. This makes the false assumption that the Gregorian calendar was in use then, but the error from this is removed when the N value for birth is removed from the N value for today (line 60). For further explanation see the previous program, Calendar.


```
 10 PRINT "BORN D-M-Y"
 20 GOSUB 110
 30 LET N1=N
 40 PRINT,,"TODAY"
 50 GOSUB 110
 60 LET N=N-N1
 70 PRINT "INTELLECT ";N-33*INT[N/33]
 80 PRINT "EMOTION ";N-28*INT[N/28]
 90 PRINT "VITALITY ";N-23*INT[N/23]
100 STOP
110 INPUT D
120 INPUT M
130 INPUT Y
140 LET N=Y*365+INT[Y/4]+INT[Y/400]-INT[Y/100]-[M<3
AND Y/4=INT[Y/4] AND [Y/100 = INT[Y/100] OR Y/400 =
INT[Y/400]]]
150 LET M$="231223345566"
160 LET N=N+CODE M$[M]+30*[M-2]+D
170 LET P=N-7*INT[N/7]+1
180 LET M$="SASUMOTUWETHFR"
190 PRINT M$[2*P-1 TO 2*P];" ";D;"-";M;"-";Y
200 RETURN
```

Bases

This program will accept a number to any base and convert it to any other base in the range 2 - 36. The limit 36 has been used because there are 36 symbols in the range 0 - 9 followed by A - Z. These symbols are used in this order so that the numerals have their normal weighting and A = 10, B = 11 up to Z = 35.

When entry and result bases have been selected, the symbol range is printed and you are invited to enter a number using those symbols. The program (which takes about a minute to work this one out) then prints out its conversion.

The accuracy and range are limited only by the way in which the computer stores floating point numbers. That is up to 10 to the power of 38 to 9 decimal places. Fractional and negative numbers should not be entered.

TECHNICAL NOTES

Lines 10 to 70 enter the numbers and print out the symbol range.

The **FOR-NEXT** loop 90 - 110 evaluates D, the decimal equivalent of the number entered in D$ to the base A.

The **FOR-NEXT** loop 130 to 170 evaluates B$ as the base B representation of the decimal number D.

Line 150 removes leading zeros.

The complicated expression in line 140 is to provide greater accuracy. Sometimes D/B**0 is evaluated as marginally less than D and so **INT** (D/B**0) gives an error in the last digit of the number B$.


```
 10 PRINT "ENTRY BASE 2-36 ";
 20 INPUT A
 30 PRINT ">>>";A,,,"RESULT BASE 2-36";
 40 INPUT B
 50 PRINT ">>>";B,,,"SYMBOL RANGE 0
TO ";CHR$[A+27]
 60 INPUT D$
 70 PRINT,,"BASE ";A;">>";D$
 80 LET D=0
 90 FOR N=LEN D$ TO 1 STEP -1
100 LET D=D+[CODE D$[N]-28]*A**[LEN D$-N]
110 NEXT N
120 LET B$=" "
130 FOR N=20+10*[B<=10] TO 0 STEP -1
140 LET B$=B$+CHR$[[N<>0]*INT[D/B**N]+[N=0]*D+28]
150 IF B$[1]="0" THEN LET B$=" "
160 LET D=D-B**N*INT[D/B**N]
170 NEXT N
180 PRINT AT 10,0;"BASE ";B;">>>";B$
```

Solitaire

This is a game for one player (surprise, surprise !). The object is to remove each peg from the board (represented here as "0" characters) leaving one peg in the centre. Pegs may be moved in any direction except diagonally. On each move a peg must jump over an adjacent peg into a vacant hole beyond, the peg passed over is then removed from the board. Play continues until a solution has been reached with just one peg left in the centre, or a stalemate situation is reached.

Use the cursor key (5 - 8) to move the flashing cursor to the piece to that you wish to move. Press "0" and the cursor "X" will stop flashing. Now press another cursor key to indicate the direction in which you wish to move. If this is legal the jump will be made, otherwise the cursor will resume flashing.

I dedicate this program to a young lady who, when shown a solution to this puzzle, said that she could do it in less moves.

TECHNICAL NOTES

Lines 10 to 50 draw the board.

Lines 70 to 80 peek the board and store in B$ the character at the cursor position on the board. The main loop 70 to 130 allows the cursor to be moved around the board.

Lines 160 to 220 detect the legality (line 200) of a proposed move, and make the move (line 210).


```
10 FOR X=2 TO 4
20 FOR Y=0 TO 6
30 PRINT AT X,Y;"O";AT Y,X;"O";AT 3,3;"▨";AT
Y,7;" "
40 NEXT Y
50 NEXT X
60 LET Y=0
70 LET C=1+X+9*Y+PEEK 16396+256*PEEK 16397
80 LET B$=CHR$ PEEK C
90 PRINT AT Y,X;"X"
100 PRINT AT Y,X;B$
110 LET X=X+[INKEY$="8" AND X<6]-[INKEY$="5" AND
X>0]
120 LET Y=Y+[INKEY$="6" AND Y<6]-[INKEY$="7" AND
Y>0]
130 IF INKEY$<>"0" THEN GOTO 70
140 PRINT AT Y,X;"X"
150 IF INKEY$<>" " THEN GOTO 150
160 PAUSE 40000
170 LET A=VAL INKEY$
180 LET X1=[A=8]-[A=5]
190 LET Y1=[A=6]-[A=7]
200 IF B$<>"O" OR PEEK [C+X1+9*Y1]<>52 OR PEEK
[C+2*X1+18*Y1]<>136 THEN GOTO 100
210 PRINT AT Y,X;"▨";AT Y+Y1,X+X1;"▨"; AT
Y+2*Y1,X+2*X1;"O"
220 GOTO 70
```

Use GRAPHICS SHIFT/H for all graphics characters in lines 30 and 210.

Minicalc

Minicalc is a very small version of the famous Visicalc program. Being only small, it will probably be more use in engineering and scientific calculations than in business forecasts. The principle behind the program is simple. You have ten formulae to calculate ten variables, A - J. Each formula may be a constant or a function of itself or other variables. Any of the following are permissible.

1.234; A + B ** **SIN**(G); H + 1;

When the program is run the message "1 - 2 - 3 A - J" appears. The options are :

1: Enter an equation.
2: Display a value and an equation
3: Calculate new values.

So if, for instance, you wished to enter an equation for variable C you would select "1C" and then enter the equation. After this you may wish to display the current value of C and its equation, so in response to "1 - 2 - 3 A - J" you would select "2C". Equations can only be 10 characters long, but the ZX81 keyword system means that functions such as **SIN,COS** which are entered as one keystroke in function mode, only count as one character. If you select "3" then the computer calculates all the new values of the variables starting with equation A. So if, for instance, equation B refers to variable C, the old value of C will be taken. Because this program uses up RAM to store the equation and values, the displayed information is minimal, and there are no checks for valid inputs, so run the program a few times to become familiar with it.

TECHNICAL NOTES

A (10) stores the values, where

A means A (1)
B means A (2)
C means A (3) etc.

A$ (10,10) stores the equations.

The heart of the program lies in lines 110 to 190 where the new values of each equation are calculated.

For each equation, the string B$ is built up from the appropiate equation in A$, but references to variables, such as B,C etc are translated to A(2), A(3) etc (line 160). When B$ is complete, the **VAL** function is applied (line 180) and the new value of the variable is found.

```
 10 DIM A[10]
 20 DIM A$[10,10]
 30 PRINT "1-2-3 A-J"
 40 INPUT B$
 50 CLS
 60 IF B$[1]="3" THEN GOTO 110
 70 LET N=CODE B$[2] - 37
 80 IF B$[1]="2" THEN PRINT A[N],,A$[N]
 90 IF B$[1]="1" THEN INPUT A$[N]
100 GOTO 30
110 FOR N=1 TO 10
120 LET B$=" "
130 FOR M=1 TO 10
140 LET C$=A$[N,M]
150 LET B$=B$+C$
160 IF C$>="A" AND C$<="J" THEN LET B$= B$[1 TO
[LEN B$ -1]]+"A["+STR$[CODE C$ -37]+"]"
170 NEXT M
180 IF B$[1]<>" " THEN LET A[N] =VAL B$
190 NEXT N
200 GOTO 30
```

Connection

This is a game about chasing words around the screen and connecting them together. First you must think of two words, which when added together give a totally new word, (this isn't strictly necessary but it makes the game more fun). Examples are: KNOW/LEDGE SAIL/OR PHOTO/GRAPH DON/ALDDUCK.

When the program is run, enter the first word (followed by newline), then the second word. You will then see the two words appear at different places on the screen. Your task now is to move the first word until it joins up with the second one. To do this you must enter a string composed of the following move characters R = Right, L = Left; U = Up; D = Down. So if, for instance, you wished to move down 3 places and right 5 places then you could enter "DDDRRRRR". If you try to move a word off the screen, the program will crash. If you don't manage to get the two words together in the first attempt the second word will take up a different position and you will have to enter a new string of moving instructions. When you succeed, your score, the number of attempts that you needed, will be displayed. Try to make the connection in the fewest attempts.

TECHNICAL NOTES

X,Y, are the co-ordinates of the first word.
X1,Y1 are the new co-ordinates.
J,K, are the co-ordinates of the second word.
A$ = first word
D$ = second word
B$ = String of move characters
Lines 140 & 150 generate X1 & Y1 from the old co-ordinates.

The FOR - NEXT loop lines 160 to 220 can either run forwards or, if the word is to be moved to the right, the loop runs backwards, moving to the last letter first.

```
 10 LET W=0
 20 LET X=0
 30 LET Y=0
 40 INPUT A$
 50 INPUT D$
 60 CLS
 70 LET J=INT[15*RND]
 80 LET  K=INT[[21-LEN[A$+D$]]*RND]+LEN A$
 90 PRINT AT Y,X;A$; AT J,K;D$
100 INPUT B$
110 LET W=W+1
120 FOR N=1 TO LEN B$
130 LET C$=B$[N]
140 LET X1=X+[C$="R"]-[C$="L"]
150 LET Y1=Y+[C$="D"]-[C$="U"]
160 IF C$<>"R" THEN FOR M=1 TO LEN A$
170 IF C$="R" THEN FOR M=LEN A$ TO 1 STEP -1
180 PRINT AT Y,X+M-1;" ";AT Y1,X1+M-1;A$[M]
190 NEXT M
200 LET X=X1
210 LET Y=Y1
220 NEXT N
230 IF X+LEN A$=K AND Y=J THEN GOTO 250
240 GOTO 60
250 PRINT "CONNECTED IN ";W;" ATTEMPTS"
```

Orbital Invaders

The main bulk of this program is written in machine code and entering it will be a very tedious business. There are over 350 bytes of hexidecimal, so I suggest that you lock yourself away where you will not be disturbed for at least an hour. There are four steps to entering the program and they may be summarised in this way :-

1. Set up the **REM** statements in lines 10 to 16.
2. Enter the code insertion program lines 100 to 150 and perform the checks given.
3. Run the program four times with different values in line 100 entering the four blocks of hex code given below. When you run the program it will ask for a string input, enter the first byte of hex (2 characters in the range 0 to F) followed by newline, this byte will then be poked into memory and also displayed on the screen. As you enter more bytes the display will scroll up.
4. Delete lines 100 to 150 and enter lines 18 to 60.

When you run the program, you will see the invaders moving from side to side across the screen (in the traditional way) firing all the time as they move. You, the defender, are just out of reach at the extreme right of the screen. You can move left by pressing "8", move right by pressing "9" and fire by pressing "0". You cannot move and fire at the same time. If you get directly underneath one of the invaders it will shoot you down and the game will automatically start again. If after a while you find the game too easy, execute the command **POKE** 16790, 255 this will make the invaders invincible, and as you can no longer shoot them down your task is simply to get to the left hand edge of the screen. May the force be with you ! !

First enter the following program into the computer :

```
10 REM WWWWWWW --177 times--WWWWWWW
12 REM XXXXXXX --96 times-- XXXXXXX
14 REM YYYYYYY --71 times-- YYYYYYY
16 REM ZZZZZZ
```

Now check that you have entered the correct no. of W,X,Y,Z characters in the following way.

The Command **PRINT CHR$ PEEK** 16690 gives "W"

16697	"X"
16792	"X"
16799	"Y"
16869	"Y"
16876	"Z"

If any of these checks give the wrong character then check your **REM** statements.

Now add the following lines.

```
100 FOR N=16514 TO 16690
110 INPUT I$
120 SCROLL
130 PRINT I$
140 POKE N,[CODE I$[1]-28]*16+CODE I$[2]-28
150 NEXT N
```

Run this and enter the following bytes of Hex.

```
FF FF FF FF FF FF FF FF FF FF FF FF 01 00 06 0C
21 82 40 36 FF 23 10 FB C9 C5 06 FF 10 FE C1 C9
3E 15 E1 BE 20 06 36 16 19 19 18 F7 3E BB 01 3C
00 BE 20 01 0E 00 C9 16 01 21 82 40 ED 5B 8F 40
16 00 19 13 7B 32 8F 40 3E 00 BE C8 2A 0C 40 01
05 00 3E 04 1D 28 0C 09 3D 20 F9 D5 11 2E 00 19
D1 18 EF ED 5B 8E 40 16 00 19 E5 3A 8F 40 D6 0D
FA 1A 41 3E 00 32 8F 40 21 8E 40 46 3A BA 40 80
32 8E 40 06 01 21 BA 40 B8 20 02 36 01 06 0D B8
20 02 36 FF E1 C3 BB 40 ED 5B BA 40 16 00 1C 1B
E1 19 36 16 23 36 07 23 36 8A 22 36 40 23 C3 EC
41
```

Change line 100 to 100 **FOR** N= 16697 **TO** 16792 and run it again entering the following codes.

2A 0E 40 11 DF FF 19 36 17 06 03 19 36 17 3E 16
19 BE 36 17 20 04 10 F3 18 0E 2B 2B 0E 04 36 16
23 0D 20 FA 36 16 2B 2B CD 9B 40 3E BB 11 21 00
36 16 19 BE 20 FA 04 05 28 57 05 CB 20 CB 20 ED
5B 8E 40 16 00 ED 52 3E 76 0E 05 BE 28 07 2B 0D
20 F9 04 18 F4 21 82 40 58 16 00 19 36 00 18 31

Change line 100 to 100 **FOR** N=16799 **TO** 16869 run the program once more entering these next codes.

01 FE EF ED 78 CB 3F 00 D2 39 41 00 2A 0E 40 36
16 CB 3F 30 0A CB 3F 38 0D 2B 3E 76 BE 20 07 23
3E 76 BE 20 01 2B 36 BB 22 0E 40 CD BB 40 11 21
00 2A 36 40 19 E5 3E 16 BE 20 06 36 15 19 19 18
F7 CD 9B 40 C3 A2 40

Now change line 100 to 100 **FOR** N= 16876 **TO** 16881 and run the program for the last time, entering 6 more codes.

36 84 23 36 16 C9

Now delete lines 100 to 150 and enter the following new lines.

```
18 CLS
20 FOR N=1 TO 32
30 PRINT "--------";
40 NEXT N
50 PRINT AT 7,31+0*USR 16528;
60 GOTO USR 16799
```

At this point I suggest you SAVE a few copies of the program, and then you are ready to run.

Mine Field

The object of this game is to traverse the minefield without being blown up. There are 10 mines planted randomly in the field and you must negotiate your way from the top left corner to the bottom right without treading on one. Fortunately you have a mine detector with you which will start registering a

mine when it is less than five moves away. The number appearing in the top left corner of the screen indicates the proximity of a mine the higher the number the nearer you are. You can move in one of four directions by pressing the cursor movement keys 5 to 8.

To make the game more difficult you can add the constraint that you must call in at the top right and bottom left corners before heading for home.

```
10 DIM A[10]
20 DIM B[10]
30 FOR N=1 TO 10
40 LET A[N]=INT[32*RND]
50 LET B[N]=INT[21*RND]
60 NEXT N
70 LET X=0
80 LET Y=1
90 PRINT AT Y,X;"*"
100 LET M=0
110 FOR N=1 TO 10
120 IF ABS[X-A[N]]<5 AND ABS[Y-B[N]]<5 THEN LET
M=M+ 10/ABS[X-A[N]+.1] + 10/ABS[Y-B[N]+.1]
130 IF X=A[N] AND Y=B[N] THEN GOTO 220
140 NEXT N
150 PRINT AT 0,0;M
160 PAUSE 40000
170 LET X=X+[INKEY$="8" AND X<31]-[INKEY$="5"
AND X>0]
180 LET Y=Y+[INKEY$"6" AND Y<20]-[INKEY$="7" AND
Y>0]
190 CLS
200 IF X=31 AND Y=20 THEN GOTO 230
210 GOTO 90
220 PRINT"BANG YOU ARE DEAD"
230 PRINT "THE END"
240 INPUT I$
250 RUN
```

Minefield Continued

TECHNICAL NOTES

A(N) and B(N) are the arrays that store the horizontal and vertical coords, respectively, of the mines.
The loop 30 to 60 positions the mines
X and Y are the coords of daring soldier.
The loop 110 to 140 evaluates M the mine detector reading for each position.
Lines 170 and 180 scan the keyboard.
At the end of the game you can press newline to start again.

Authors Advice

This section contains a few tips for people wishing to write their own programs for the 1K ZX81.

1. Before you start programming have a fairly clear idea of what you wish to achieve and do a few rough calculations to see whether there is any chance of squeezing it into 1K.

2. If you wish to move just a few objects around the screen, clear the screen between each move redrawing the objects as you go. This will save on display file memory.

3. When debugging programs using subroutines beware of leaving old return addresses on the gosub stack by stopping the program from within a subroutine. Executing **RETURN** as a command (no linenumber) will clear off one address.

4. Re-use variables wherever possible.

5. If you wish to store numerical data, rather than declaring a lot of numerical variables, and using up program space assigning values to them, try putting the data into a string array. See CALENDAR.

6. If all else fails admit defeat and buy a RAM PACK.

£4.95