

The book cover features a light blue background with a pattern of yellow wavy lines. Several puzzle pieces are scattered across the cover: three dark green pieces and one yellow piece. The yellow piece in the bottom right corner contains the text 'SHIVA'S micro puzzle books'.

COMPUTER PUZZLES: FOR SPECTRUM & ZX81

Ian Stewart and Robin Jones

SHIVA'S
micro puzzle books

**COMPUTER PUZZLES:
FOR SPECTRUM & ZX81**

COMPUTER PUZZLES: FOR SPECTRUM & ZX81

Ian Stewart

Mathematics Institute, University of Warwick

Robin Jones

Computer Unit, South Kent College of Technology



Shiva Publishing Limited

SHIVA PUBLISHING LIMITED
4 Church Lane, Nantwich, Cheshire CW5 5RQ, England

© Ian Stewart and Robin Jones, 1982

ISBN 0 906812 27 5

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, electronic, mechanical, photocopying, recording and/or otherwise, without the prior written permission of the Publishers.

This book is sold subject to the Standard Conditions of Sale of Net Books and may not be resold in the UK below the net price given by the Publishers in their current price list.

Typeset and printed by Devon Print Group, Exeter

contents

Introduction	3
Read this bit first . . .	4
1 The wolf, the goat and the cabbage	5
2 Alphabetical disorder	10
3 Cat-swap	13
4 Stringing the ton	15
5 Fair shares	18
6 The magic forest	21
7 Queens dominant	24
8 Bishops blatant	27
9 Rooks rampant	28
10 Shy ladies	29
11 Monkey puzzle	30
12 Ecological chain	33
13 Tower of Hanoi with pancakes	36
14 Several times knightly	39
15 Beasts on the steps	43
16 Solitaire	48
17 Anagrams against the clock	51
18 Hidden words	54
19 Cryptograms	58

introduction

The great puzzle-composer Henry Ernest Dudeney once wrote: “The history of puzzles entails nothing short of the actual story of the beginnings and development of exact thinking in Man. If there were no puzzles to solve, there would be no questions to ask; and if there were no questions to be asked, what a dull world it would be!”

Now, Dudeney made his living out of puzzles, so we may forgive any slight excesses; but it is probably true that puzzles tend to encourage a certain amount of logical thinking. “Problem-solving” is one of the important arts of our age.

But above all, puzzles are *fun*.

There is a long tradition of puzzles, going back to ancient Babylon and beyond. Many of the puzzles are as fascinating now as they were seven thousand years ago. And, to prove it, puzzles now enter the Computer Age. This is a book of computer puzzles, to run on the Sinclair ZX Spectrum and ZX81. Computers are excellent for *experimenting* with puzzles: you can try an answer and see whether it works.

Some puzzles, in fact, can *only* be made to work on a computer. For instance, you could computerize a four-dimensional Rubik hypercube; but you could never *make* one. (Unfortunately you probably couldn't *solve* one either, so I haven't pursued this particular idea here.) On a colour computer you can change colours in ways which would be very tricky to achieve with the traditional “hardware” of pencil and paper, tiddlywinks and chessboards.


Other puzzles adapt well. You know the one about the wolf, the goat, and the cabbage, trying to cross the river? On a computer you can draw the river, load the boat, watch it float across the river, and see who gets eaten . . .

Anyway, this book has three purposes: to move puzzles into the Computer Age; to teach a bit of clear thinking; but mostly, for enjoyment. It was fun writing it: I hope you have as much fun reading it and running the programs.

read this bit first...

... or you'll get in a muddle.

1. Most programs in this book come in two versions: the first for the ZX81 (with at least 4K of RAM), the second for the ZX Spectrum. The last four programs are in a special section for Spectrum enthusiasts.
2. The ZX81 version will of course *run* on a Spectrum, but the Spectrum version makes use of the extra facilities available.
3. In order to save space, hence paper, hence save you money, the Spectrum listing is found by modifying the ZX81 listing. The Spectrum section shows exactly how to do this, in full detail. Any ZX81 lines that will be changed in a Spectrum version are marked with an arrow → to save Spectrum-owners time in entering the programs.
4. One nuisance: ZX81 variables are all upper-case (A, B, C, etc.); but Spectrum variables are normally taken as lower-case (a, b, c, etc.). Again, to save paper, the Spectrum version is *not* rewritten to lower the case of all variables; but it is assumed that when you type the program in, you will in fact use *lower* case. So PRINT X, Y in the listing must be entered by Spectromaniacs as PRINT x, y . . . etc.
5. A box □ is used to represent a space when this is not otherwise obviously needed, ■ is an inverse video space, and boxes around characters denote inverse video.
6. ZX81 he say "GOTO" and "GOSUB"; whereas Spectrum he say "GO TO" and "GO SUB". No notice is taken of this in the listings.
7. A particular problem for typesetters is the use of graphics characters. The conventions in this book are:

ZX81: Small g means "graphics character": it is followed by the letter or number of the right key. So "gT" is .

Spectrum: Similarly; but characters accessed via CAPS SHIFT get a "c" added on the end. So "g6" is ; and "g6c" is .

8. On the Spectrum, you should set up BORDER, PAPER, and INK colours directly from the keyboard, before running each program. To do this, just enter in command mode something like
BORDER 1: PAPER 3: INK 7
for a blue border, magenta paper, and black ink. (Hit ENTER twice to get the PAPER colour showing.) Any combination is in principle permitted, but some look rather yucky. Experiment; you can always change the colours before a RUN. I have *not* included these initial commands in the Spectrum programs, since they are so routine; most Spectrum versions introduce extra colour commands anyway.
9. The programs are ready to use on their own, and need not be run through in the order they occur in the book: dip in as it pleases you, copy the listing in, and follow the instructions.
10. Enough of this idle chip-chat (*sic*)—to work!

Ferry your possessions across the river—but be careful! They eat each other.

1 the wolf, the goat and the cabbage

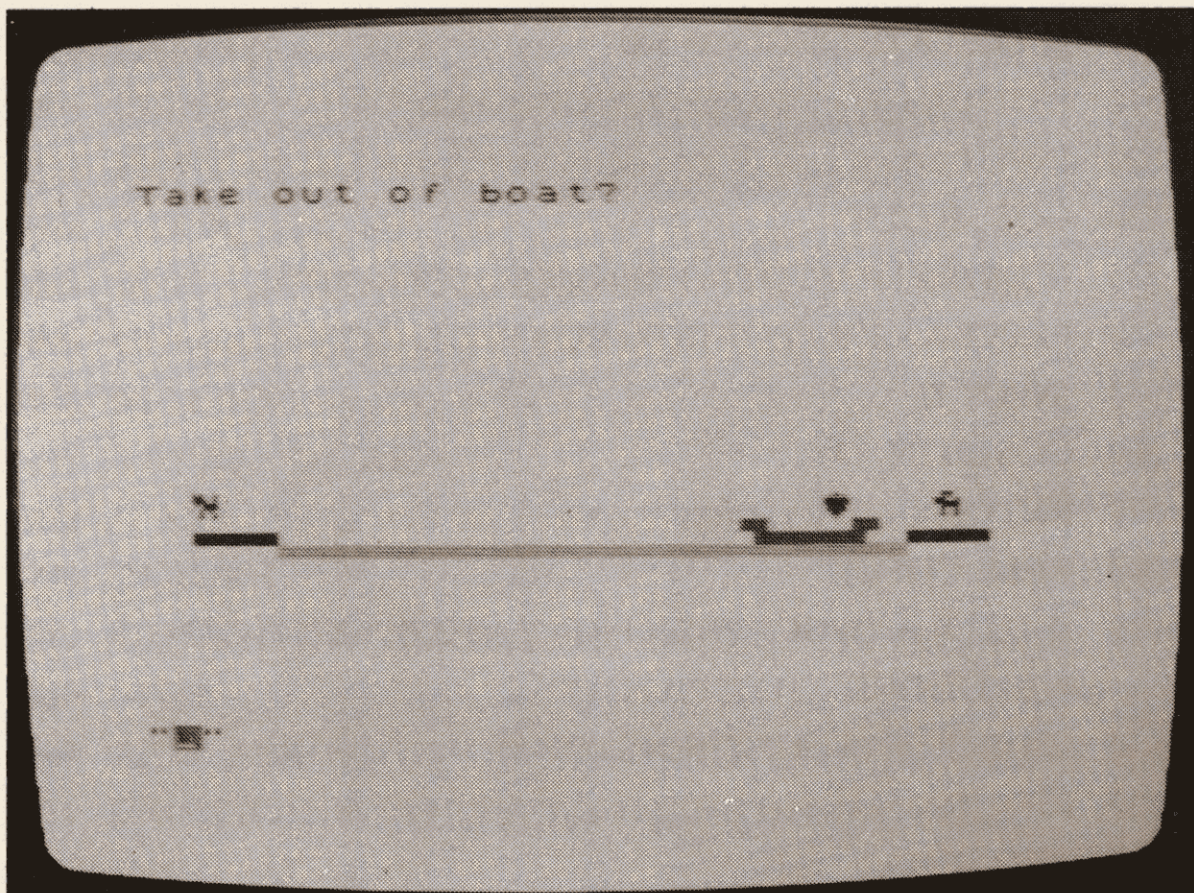
Imagine that you are a seventeenth-century farmer, taking his wares to market. They are:

- 1 wolf
- 1 goat
- 1 cabbage.

Don't ask me the whys and wherefores of wolf-farming, I'm only the author. You come to a river, which you must cross. There is a boat, which will carry you, together with at most one additional item (wolf, goat, or cabbage).

While you are in the vicinity, your wares behave their little selves admirably. *But:* should you leave the wolf alone with the goat on one bank, while you are crossing in the boat or sitting on the far side . . . well, wolves are rather partial to a nice bit of goat steak. And, similarly, goats can nibble a pretty big hole in your prize cabbage. Luckily, the wolf detests cabbage (he went to boarding-school as a young cub) and the cabbage is a very peaceable fellow indeed; so you can leave these two together if you wish.

What items must be moved, and in what order, to cross the river?



ZX81 VERSION

```
→ 10 LET B$ = "□ gR gF gF gE □"
20 LET BC = 0
30 LET QUERY = 500
40 LET CHECK = 1000
50 LET BOAT = 2000
60 LET BK = 1
70 LET OUT = 2500
80 LET IN = 3000
82 LET ERROR = 5000
85 LET EAT = 4000
90 LET C$ = "□□□□□□□"
93 DIM D$(2, 3)
→ 95 LET D$(1) = "WGC"
→ 100 PRINT AT 15, 0; "□□ g6 g6 g6 [22 spaces] g6 g6 g6"
→ 110 PRINT AT 16, 0; "□□ ■■■ gS [22 times] ■■■"
120 PRINT AT 14, 2; D$(1)
→ 130 PRINT AT 15, 5; B$
200 REM MOVE
210 GOSUB QUERY
220 GOSUB BOAT
230 GOTO 200
500 REM QUERY
505 IF BC = 0 THEN GOTO 540
510 PRINT AT 2, 0; "TAKE OUT OF BOAT?"
520 INPUT O$
530 GOSUB OUT
535 IF D$(BK) = "□□□" THEN GOTO 570
540 PRINT AT 2, 0; "PUT INTO BOAT? □□□□"
550 INPUT I$
560 GOSUB IN
570 RETURN
1000 REM CHECK
→ 1010 IF D$(2) = "WGC" THEN PRINT AT 2, 0; "□□□□□□□
WELL DONE □□□□□□□"
→ 1020 IF D$(2) = "WGC" THEN STOP
1030 RETURN
2000 REM BOAT
2010 FOR T = 5 * (BK = 1) + 20 * (BK < > 1) TO 20 * (BK = 1)
+ 5 * (BK < > 1) STEP 1 - 2 * (BK < > 1)
```

```

→ 2020 PRINT AT 15, T; B$
2025 PRINT AT 14, T; C$
2030 NEXT T
2035 GOSUB EAT
2040 LET BK = 3 - BK
2050 RETURN
2500 REM OUT
2505 IF O$ = " " THEN RETURN
2510 FOR F = 3 TO 5
→ 2520 IF C$(F) = O$ THEN GOTO 2540
2530 NEXT F
2540 LET C$(F) = "□"
→ 2550 LET D$(BK, F - 2) = O$
2560 PRINT AT 14, 2 * (BK = 1) + 27 * (BK < > 1); D$(BK)
2570 PRINT AT 14, 5 * (BK = 1) + 20 * (BK < > 1); C$
2580 LET BC = BC - 1
2590 GOSUB CHECK
2600 RETURN
3000 REM IN
3005 IF I$ = " " THEN RETURN
3010 FOR F = 1 TO 3
→ 3020 IF D$(BK, F) = I$ THEN GOTO 3040
3030 NEXT F
3040 LET D$(BK, F) = "□"
→ 3050 LET C$(F + 2) = I$
3060 PRINT AT 14, 2 * (BK = 1) + 27 * (BK < > 1); D$(BK)
3070 PRINT AT 14, 5 * (BK = 1) + 20 * (BK < > 1); C$
3080 LET BC = BC + 1
3085 IF BC = 2 THEN GOTO ERROR
3090 GOSUB CHECK
4000 REM EAT
4010 LET X$ = D$(BK)
→ 4020 IF X$ = "WG□" OR X$ = "□GC" THEN GOTO 4040
4030 RETURN
4040 PRINT AT 2, 0; "□□□□□□□ WOOPS...EATEN □□□□□□"
4050 STOP
5000 REM ERROR
5010 PRINT AT 2, 0; "SORRY, TOO MANY IN BOAT. TRY"
5020 PRINT "AGAIN"
5030 STOP

```

USING THE ZX81 VERSION

Type it all in, *check* it; press RUN. You will see the river; the boat; and the Wolf, Goat, and Cabbage represented by their initials, sitting on the bank. (I haven't drawn you in, but you're there: you move with the boat.)

At each stage you will be asked what, if anything, you wish to take *out of* the boat. To this you may answer "W", "G", "C"—if they're in there to take out—or "□" if you don't want anything removed. Next you are asked for what goes *into* the boat: there is the same choice of inputs.

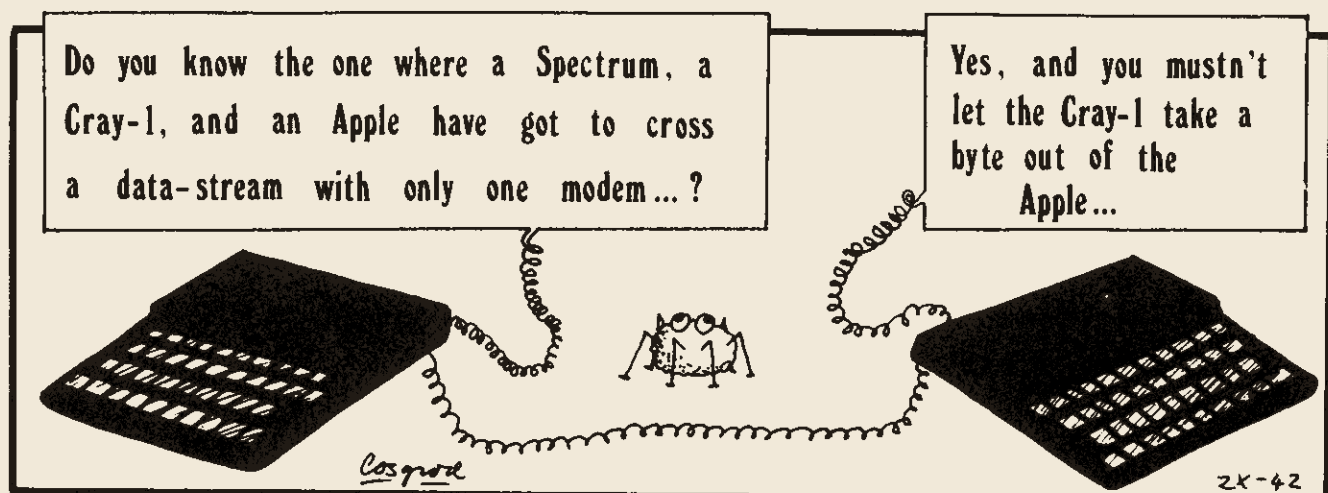
The boat then starts to cross the river, and you soon find out if you made a good choice.

If you get everything across, the computer says so.

ZX SPECTRUM VERSION

Change the following lines:

```
10 LET B$ = "□ g7 g3c g3c g3c g4c □"  
95 LET D$(1) = "gA gG gC"  
100 PRINT AT 15, 0; "□ □ g3c g3c g3c [22 spaces] g3c g3c g3c"  
110 PRINT AT 16, 0; "□ □ □ □ □"; INK 5; "g3 [22 times] "; INK 0; "□ □ □"  
130 PRINT AT 15, 5; INK 3; B$  
1010 IF D$(2) = "gA gG gC" THEN PRINT AT 2, 0; "□ □ □ □ □ □ □ □  
    Well done! □ □ □ □ □ □ □": STOP  
1020 [delete this line altogether]  
2020 PRINT INK 3; AT 15, T; B$  
2520 IF CODE C$(F) = CODE O$ + 47 THEN GOTO 2540  
2550 LET D$(BK, F - 2) = CHR$(47 + CODE O$)  
3020 IF CODE D$(BK, F) = CODE I$ + 47 THEN GOTO 3040  
3050 LET C$(F + 2) = CHR$(47 + CODE I$)  
4020 IF X$ = "gA gG □" OR X$ = "□ gG gC" THEN GO TO 4040
```



Add these lines:

```
525 IF O$ = "W" THEN LET O$ = "A"  
555 IF I$ = "W" THEN LET I$ = "A"  
2045 INK 0
```

Now add lines to set up three user-defined graphics for the Wolf, Goat, and Cabbage (at A G C in the user-defined graphics character section):

```
6000 LET Y$ = "AGC"  
6010 DATA 162, 226, 242, 94, 30, 36, 34, 34, 56, 40, 226, 254, 62, 34, 34, 34, 16,  
84, 254, 254, 124, 124, 56, 16  
6020 FOR X = 1 TO 3  
6030 FOR N = 0 TO 7  
6040 READ M  
6050 POKE USR Y$ (X) + N, M  
6060 NEXT N  
6070 NEXT X
```

If you want some sound-effects for a solution, also add lines:

```
1015 IF D$ (2) = "gA gG gC" THEN BEEP .3, 0: BEEP .3, 4: BEEP .3, 7:  
BEEP 1.5, 10: STOP
```

And delete:

```
STOP in 1010
```

USING THE SPECTRUM VERSION

1. Start, on first switching on, with GOTO 6000 to set up the graphics characters.
2. Thereafter, restart the program with RUN; proceed as in the ZX81 version.

*Shuffle the alphabet (missing Y and Z)
back into shape:*

2 alphabetical disorder

This program starts with the alphabet arranged like this:

```
A F K P U
B G L Q V
C H M R W
D I N S X
E J O T □
```

and the first thing it does is jumble them all up.

Your job is to get them back in order again. You do this by “moving” any letter adjacent to the blank space □, into the space. Of course the space now ends up where the letter was. In other words, you imagine yourself moving the space around the rectangle of letters.

To move the space, use the four “arrow” keys 5, 6, 7, 8: the arrow gives the direction.

ZX81 VERSION

```
10 FOR I = 1 TO 6
→ 20 PRINT AT 2 * I, 10; “gH gH gH gH gH gH gH gH gH gH gH”
30 NEXT I
40 FOR I = 1 TO 5
→ 50 PRINT AT 2 * I + 1, 10; “gH□gH□gH□gH□gH□gH”
60 NEXT I
70 DIM A$(5, 5)
100 FOR I = 1 TO 5
110 FOR J = 1 TO 5
→ 115 LET A$(I, J) = CHR$(32 + I + 5 * J - 62 * (I = 5 AND J = 5))
120 PRINT AT 2 * I + 1, 2 * J + 9; A$(I, J)
130 NEXT J
140 NEXT I
```



```

200 LET P = 5
210 LET Q = 5
215 FOR T = 1 TO 50
220 LET W = 2 * INT ( 2 * RND ) - 1
230 LET P0 = P + W
240 IF P0 > 5 OR P0 < 1 THEN GOTO 220
250 LET W = 2 * INT ( 2 * RND ) - 1
260 LET Q0 = Q + W
270 IF Q0 > 5 OR Q0 < 1 THEN GOTO 250
275 GOSUB 290
280 NEXT T
285 GOTO 400
290 LET Y$ = A$ (P0, Q0)
300 LET A$ (P0, Q0) = A$ (P, Q)
310 LET A$ (P, Q) = Y$
320 PRINT AT 2 * P0 + 1, 2 * Q0 + 9; A$ (P0, Q0)
330 PRINT AT 2 * P + 1, 2 * Q + 9; A$ (P, Q)
340 LET P = P0
350 LET Q = Q0
360 RETURN
400 PRINT AT 21, 0; "NOW IT " " S YOUR TURN"
410 IF INKEY$ < > " " THEN GOTO 410
420 IF INKEY$ = " " THEN GOTO 420
430 LET I$ = INKEY$
440 IF I$ = "5" THEN LET Q0 = Q - 1
450 IF I$ = "6" THEN LET P0 = P + 1
460 IF I$ = "7" THEN LET P0 = P - 1
470 IF I$ = "8" THEN LET Q0 = Q + 1
480 IF P0 < 1 THEN LET P0 = 1: GOTO 410
481 IF P0 > 5 THEN LET P0 = 5: GOTO 410
482 IF Q0 < 1 THEN LET Q0 = 1: GOTO 410
483 IF Q0 > 5 THEN LET Q0 = 5: GOTO 410
490 GOSUB 290
500 GOTO 410

```

USING THE ZX81 VERSION

Start with RUN; wait for the jumbling to finish; now start pressing those arrow keys!

SPECTRUM VERSION

Change the following lines:

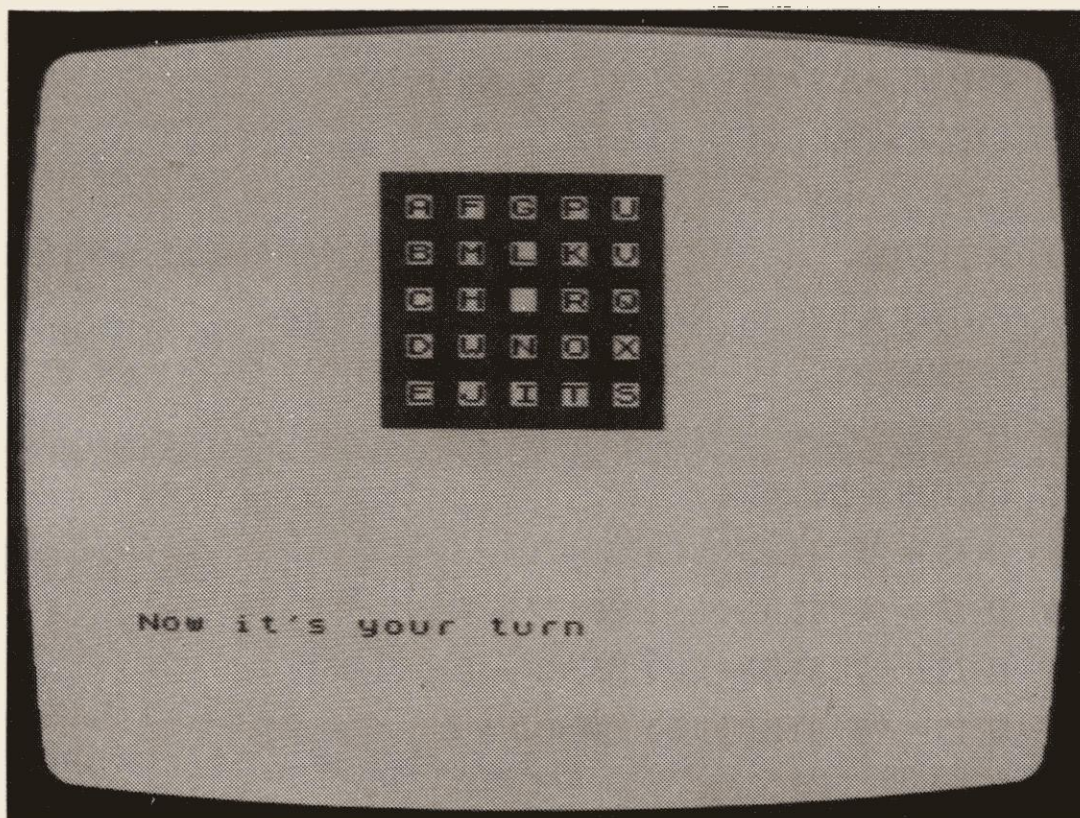
```
20 PRINT AT 2 * I, 10; PAPER 3; " [11 spaces] "  
50 PRINT AT 2 * I + 1, 10; INK 3; "■□■□■□■□■□■"  
115 LET A$(I, J) = CHR$(I + 5 * J + 59 - (I = 5 AND J = 5) * 57)
```

Add the line:

```
355 BEEP .1, P + 5 * Q
```

USING THE SPECTRUM VERSION

Start with RUN; wait (rather less time) for the jumbling to finish; start pressing the arrow keys.



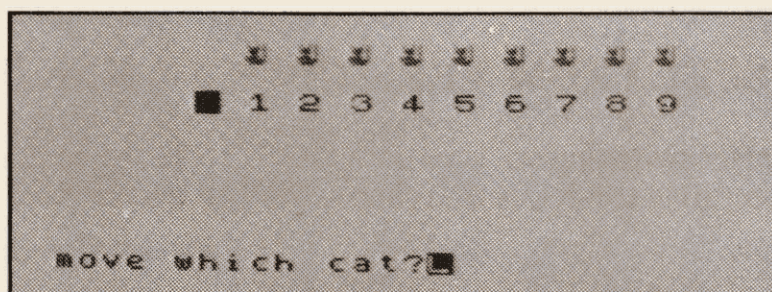
*A line of cats sits on a wall.
Can you swap them round?*

3 cat-swap

The program presents you with a line of numbered cats, in order 1 2 3 4 5 6 7 (say). You must rearrange them into order 7 6 5 4 3 2 1. There is one blank space on the left-hand end. A cat may move into the space if it is adjacent to it; or it may hop over *exactly one* cat (like a draughts piece taking) into the vacant space. You choose which cat to move, and keep going until they are in order. The space must end up where it started out, at the left.

ZX81 VERSION

```
10 LET P = 0
20 DIM W (9)
30 FOR I = 1 TO 9
40 LET W (I) = I
50 NEXT I
→ 100 PRINT AT 0, 0; "HOW MANY CATS"
→ 105 INPUT C
110 LET N$ = " 1 2 3 4 5 6 7 8 9"
120 PRINT AT 16, 6; "■" + N$ (TO 2 * C)
130 FOR I = 1 TO C
→ 140 PRINT AT 14, 6 + 2 * I; "*"
150 NEXT I
→ 200 PRINT AT 0, 0; "MOVE WHICH CAT"
→ 205 INPUT A
210 IF A < 1 OR A > C THEN GOTO 200
212 PRINT AT 0, 0; " [14 spaces] "
```



```

215 IF ABS (W (A) - P) > 2 THEN GOTO 200
→ 220 PRINT AT 16, 6 + 2 * P; A; AT 14, 6 + 2 * P; "*"; AT 16, 6 + 2 * W (A);
    "■"; AT 14, 6 + 2 * W (A); "□"
224 LET R = W (A)
225 LET W (A) = P
230 LET P = R
240 FOR J = 1 TO C
250 IF W (J) + J < > C + 1 THEN GOTO 200
260 NEXT J
265 IF P > 0 THEN GOTO 200
270 PRINT AT 8, 12; "CATS SWAPPED"
280 STOP

```

USING THE ZX81 VERSION

Type RUN. The cats are the asterisks. Choose the numbers, see how they jump. If you get them in the right order, the computer says so.

SPECTRUM VERSION

Change the following lines:

```

100 INPUT "How many cats?"; C
105 [delete this line]
140 PRINT AT 14, 6 + 2 * I; INK 4; "gC"; INK 0
200 INPUT "Move which cat?"; A
205 [delete this line]
220 PRINT AT 16, 6 + 2 * P; A; AT 14, 6 + 2 * P; INK 4; "gC"; INK 0;
    AT 16, 6 + 2 * W (A); "■"; AT 14, 6 + 2 * W (A); "□": BEEP .1, A

```

Add a routine to set up the graphics character:

```

500 DATA 74, 122, 122, 50, 250, 254, 124, 0
510 FOR T = 0 TO 7
520 READ X
530 POKE USR "C" + T, X
540 NEXT T

```

USING THE SPECTRUM VERSION

First time around after switching on, start with GO TO 500. When the graphics character has been set up, you may use RUN thereafter. Choose which cat to hop, just as in the ZX81 version.

Use the nine digits to make 100:

9 stringing the ton

The screen will show you the nine digits 1–9, with spaces between them; and an arrow pointing to a space. You can move the arrow and insert in the spaces any of the signs +, −, *, /; or remove them if you decide to later. When you think what is on the screen works out to 100 press “P” for a check.

Gaps between digits are ignored when working out the answer; for instance if you’d reached

$$1 + 2 \quad 3 + 4 \quad 5 + 6 + 7 + 8 - 9$$

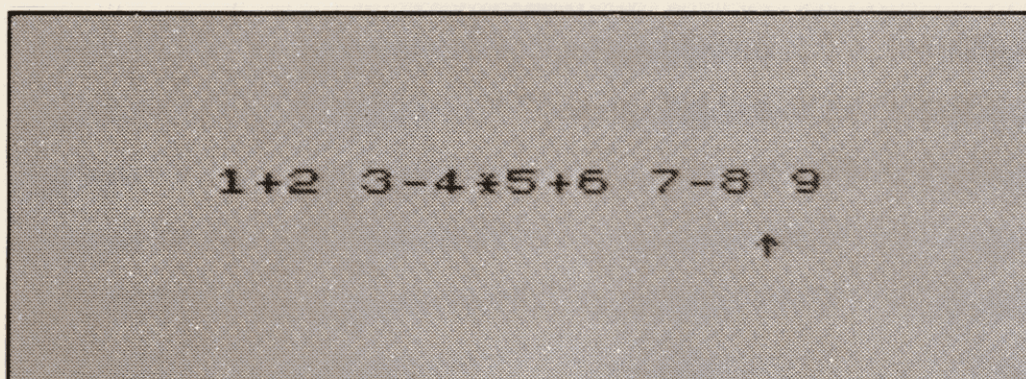
this is interpreted as

$$1 + 23 + 45 + 6 + 7 + 8 - 9$$

which is

$$81.$$

Close-ish, but not good enough: try again!



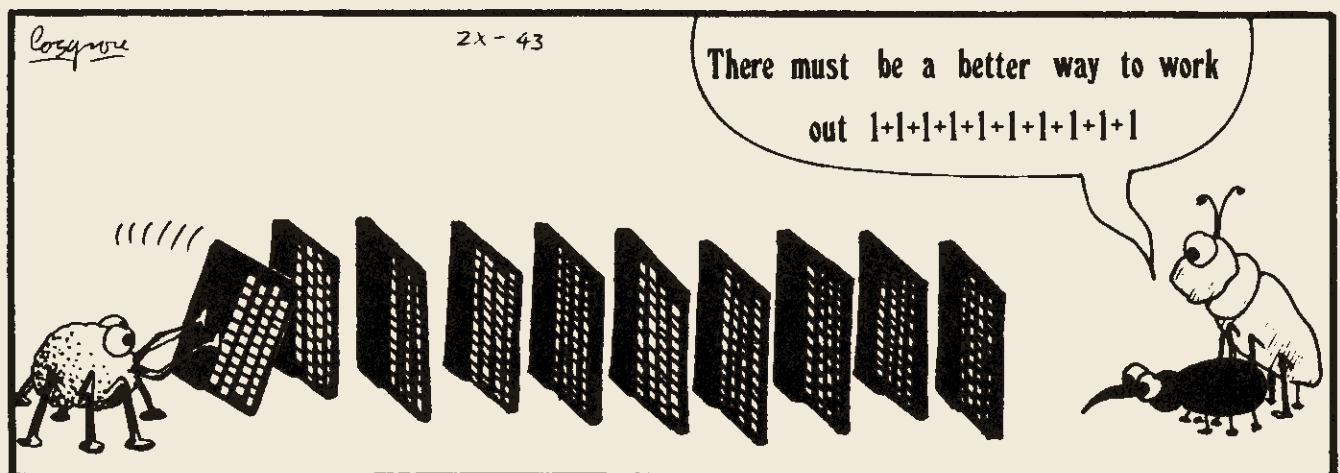
ZX81 VERSION

```
10 LET N$ = "1□2□3□4□5□6□7□8□9"  
→ 20 LET C$ = "□□>□□"  
30 LET H = 1  
100 PRINT AT 16, 10; N$  
110 PRINT AT 18, 7 + 2 * H; C$  
120 IF INKEY$ <> " " THEN GOTO 120  
130 IF INKEY$ = " " THEN GOTO 130  
140 LET I$ = INKEY$
```

```

150 IF I$ = "5" THEN GOSUB 300
160 IF I$ = "8" THEN GOSUB 400
170 IF I$ = "B" THEN LET N$ (2 * H) = "*"
180 IF I$ = "K" THEN LET N$ (2 * H) = "+"
190 IF I$ = "J" THEN LET N$ (2 * H) = "-"
200 IF I$ = "V" THEN LET N$ (2 * H) = "/"
210 IF I$ = "X" THEN LET N$ (2 * H) = "□"
220 IF I$ = "P" THEN GOTO 1000
230 GOTO 100
300 LET H = H - 1 + (H = 1)
310 PRINT AT 18, 7 + 2 * H; C$
320 RETURN
400 LET H = H + 1 - (H = 8)
410 GOTO 310
1000 LET A$ = " "
1010 FOR T = 1 TO 17
1020 IF N$ (T) < > "□" THEN LET A$ = A$ + N$ (T)
1030 NEXT T
1040 PRINT AT 1, 5; "THE VALUE IS □"; VAL A$
→ 1050 IF VAL A$ = 100 THEN PRINT AT 3, 5; "WELL DONE"
1060 IF VAL A$ = 100 THEN STOP
1070 FOR J = 1 TO 50
1080 NEXT J
1090 PRINT AT 1, 0; "[32 spaces]"
1100 PRINT AT 3, 0; "[32 spaces]"
1110 GOTO 100

```



USING THE ZX81 VERSION

RUN; use the keys 5 and 8 to move the cursor; input arithmetical signs using the keys they are written on; input a space using the X key. When you are happy, press P: see if you've reached the century.

SPECTRUM VERSION:

Change the following lines:

```
20 LET C$ = "□□↑□□"  
1050 IF VAL A$ = 100 THEN PRINT AT 3, 5; "Well done!": BEEP .3, 0:  
      BEEP .3, 4: BEEP . 3, 7: BEEP 1.5, 10
```

USING THE SPECTRUM VERSION

Just as for the ZX81.

*Divide the wine, but don't
spill it!*

5 fair shares

You have three glasses of claret (this is the SDP version) which hold respectively 8, 5, and 3 measures. The 8-measure glass is full, the others empty. You must divide the wine into two exactly equal portions, by pouring from one glass to another. You may only stop pouring when one glass gets full or the other gets empty.

Can you do it?

This is a famous problem; it's been around at least since 1400 AD. The other famous problem has been getting your hands on the bottle to begin with . . .

ZX81 VERSION

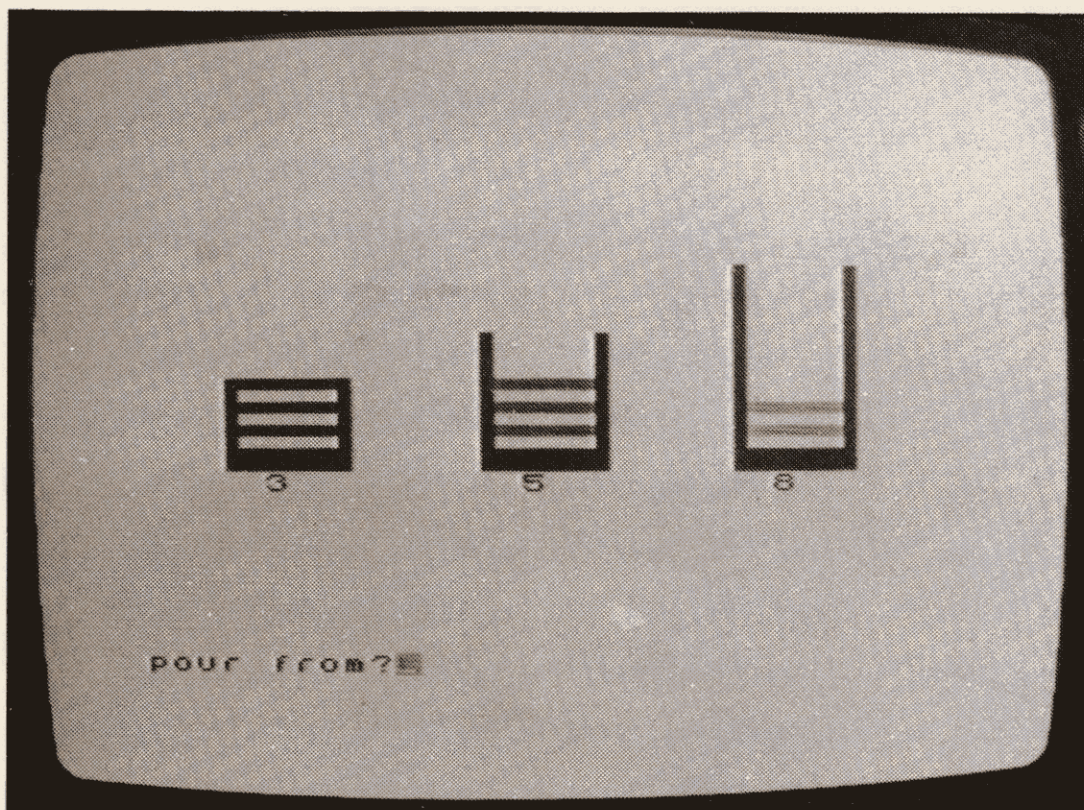
```
10 LET A$ = "□□□□"  
→ 20 LET B$ = "gG gG gG gG"  
→ 30 LET C$ = "g8 □□□□ g5"  
→ 40 LET D$ = "g8 □□□□ g5"  
50 DIM H (3)  
60 LET H (3) = 8  
100 FOR I = 1 TO 3  
110 LET K = 3 * I - (I > 1)  
120 FOR T = 1 TO K  
130 PRINT AT 13 - K + T, 10 * I - 7; C$  
140 NEXT T  
150 PRINT AT 14, 10 * I - 7; D$  
160 PRINT AT 15, 10 * I - 5; K  
170 NEXT I  
180 GOSUB 1000  
→ 200 PRINT AT 0, 0; "POUR FROM?"  
→ 205 INPUT F  
207 IF F < > 3 AND F < > 5 AND F < > 8 THEN GOTO 200  
→ 210 PRINT AT 0, 0; "POUR INTO?"  
→ 215 INPUT G
```



```

217 IF G <> 3 AND G <> 5 AND G <> 8 THEN GOTO 210
220 LET I = INT ( (F + 1) / 3)
230 LET J = INT ( (G + 1) / 3)
240 LET Q = H (I) + H (J)
300 IF Q < G THEN GOSUB 400
310 IF Q <= G THEN GOSUB 500
320 GOSUB 1000
330 IF H (2) = 4 AND H (3) = 4 THEN GOSUB 600
340 GOTO 200
400 LET H (J) = G
410 LET H (I) = Q - G
420 RETURN
500 LET H (J) = Q
510 LET H (I) = 0
520 RETURN
600 PRINT AT 2, 10; "WELL DONE"
610 STOP
1000 FOR I = 1 TO 3
1010 FOR T = 1 TO H (I)
1020 PRINT AT 14 - T, 10 * I - 6; B$
1030 NEXT T
1040 FOR T = H (I) + 1 TO 3 * I - (I > 1)
1050 PRINT AT 14 - T, 10 * I - 6; A$
1060 NEXT T
1070 NEXT I
1090 RETURN

```



USING THE ZX81 VERSION

RUN; Choose the glass to pour from and into, using the numbers written below them; repeat until you get 4 measures in glass 5 and 4 in glass 8.

SPECTRUM VERSION

Change the following lines:

```
20 LET B$ = "g3 g3 g3 g3"  
30 LET C$ = "g5 □ □ □ □ g5c"  
40 LET D$ = "g5 g8c g8c g8c g8c g5c"  
200 INPUT "Pour From?"; F  
205 [delete this line]  
210 INPUT "Pour Into?"; G  
215 [delete this line]
```

Add these:

```
70 LET G = 8  
1005 INK 2 + I  
1080 INK 0  
1025 IF I = INT ( (G + 1) / 3) THEN BEEP .05, 5 - T:  
      BEEP .05, 5 - T - .3: BEEP .05, 5 - T - .7
```

USING THE SPECTRUM VERSION

As for the ZX81. The wine changes colour . . .

*As you move through the forest,
you accumulate a magic charge:
to emerge it has to be zero.
Can you get through?*

6 the magic forest

Here the computer draws a square forest, with numbered squares. This number represents the amount of "magic charge" at that point in the forest. You move a cursor through the forest, aiming to get to sanctuary on the other side: sanctuary is the goalpost-shaped area. As you encounter a given square its magic charge *adds* to yours. However, being magic, the adds ignore carries; so for instance $5 + 7$ is 2, not 12; and $8 + 6$ is 4. That way, you can lose magic charge, because (say) $3 + 7 = 0$ and the charge has all gone.

Can you emerge from the wood with zero magic charge?

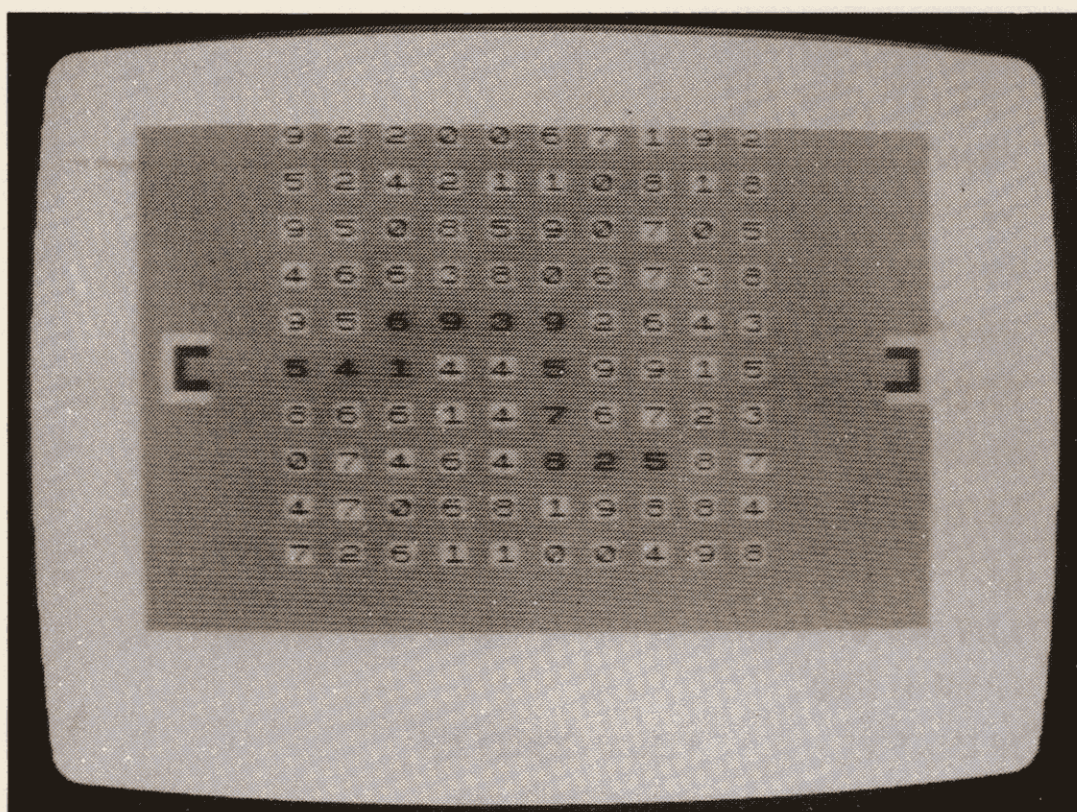
ZX81 VERSION

```
→ 1 FAST
   2 LET C = 0
   5 DIM D (10, 10)
  10 FOR I = 1 TO 10
  20 FOR J = 1 TO 10
  30 LET D (I, J) = INT (10 * RND)
  35 PRINT AT 2 * I - 2, 2 * J + 4; D (I, J)
  40 NEXT J
  50 NEXT I
→ 60 PRINT AT 9, 1; "g3 g6"; AT 10, 1; "g8"; AT 11, 1; "g2 g7"
→ 70 PRINT AT 9, 30; "g6 g4"; AT 10, 31; "g5"; AT 11, 30; "g7 g1"
→ 80 SLOW
 100 LET P = 10
 105 LET Q = 2
 110 LET N = 0
 120 GOSUB 1000
 200 IF INKEY$ < > " " THEN GOTO 200
 210 IF INKEY$ = " " THEN GOTO 210
```

```

220 LET I$ = INKEY$
224 LET P0 = P
226 LET Q0 = Q
230 IF I$ = "5" THEN LET Q = Q - 2
240 IF I$ = "6" THEN LET P = P + 2
250 IF I$ = "7" THEN LET P = P - 2
260 IF I$ = "8" THEN LET Q = Q + 2
262 LET C = C + 1
265 IF P > 18 THEN GOTO 500
270 IF Q < 6 OR Q > 24 THEN GOTO 300
280 LET N = N + D (1 + P/2, Q/2 - 2)
290 IF N > 9 THEN LET N = N - 10
300 GOSUB 1000
310 GOSUB 1020
320 IF P = 10 AND Q = 30 THEN GOTO 2000
400 GOTO 200
500 LET P = P - 2
510 PRINT AT 21, 0; "OUT OF BOUNDS"
520 FOR T = 1 TO 50
530 NEXT T
540 PRINT AT 21, 0; "□□□□□□□□□□□□□□"
550 GOTO 200
→ 1000 PRINT AT P, Q; CHR$(156 + N)
1010 RETURN

```



```

1020 IF Q0 < 6 OR Q > = 24 THEN PRINT AT P0, Q0: "□"
1030 IF Q0 > 5 AND Q0 < 25 THEN PRINT AT P0, Q0:
      D (1 + P0/2, Q0/2 - 2)
1040 RETURN
2000 IF N= 0 THEN PRINT AT 20, 0: "CONGRATULATIONS.
      YOU TOOK □".., C: "□ MOVES"
2010 IF N < > 0 THEN PRINT AT 20, 0: "SORRY. WRONG TOTAL - "..
      "PLEASE TRY AGAIN."

```

USING THE ZX81 VERSION

RUN; wait for the screen picture to appear. The wood is in the middle; you're on the left, sitting inside a "goalpost" area. There is another goalpost on the far side, and you have to end up snugly inside that one. The numbers on the forest and you are the magic charges.

Using the four arrow keys, you can move at will across the screen. Watch how the numbers add as you move. If you choose your path carefully, you can come out with charge zero.

The program tells you how many moves you needed. See how few you can manage it in!

SPECTRUM VERSION

Change the following lines:

```

1 PAPER 7: INK 0: CLS
60 PRINT AT 9, 1: "g4 g3c"; AT 10, 1: "g5"; AT 11, 1: "g1 g3"
70 PRINT AT 9, 30; "g3c g7c"; AT 10, 31; "g5c"; AT 11, 30; "g3 g2"
80 [delete this line]
1000 PRINT AT P, Q: FLASH 1: CHR$(48 + N)

```

Get some colour into it by adding:

```

8 PAPER 4
55 INK 2
75 INK 0
77 PAPER 4

```

And some sound with

```

222 BEEP .3, 2 * VAL I$

```

USING THE SPECTRUM VERSION

The same as for the ZX81.

*How many queens do you need
to attack the whole chessboard?*

7 queens dominant

This program lets you distribute chess queens on a board, and shows which squares they are attacking. The aim is to attack all of the board with as few queens as possible, by judicious choice. It *can* be done with five of them.

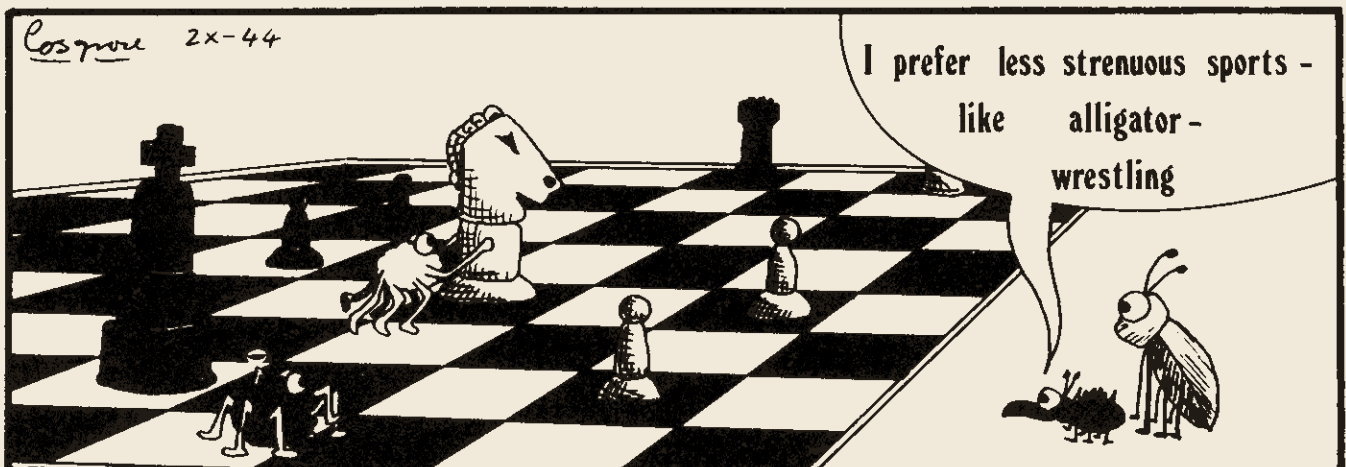
ZX81 VERSION

```
10 DIM B (8, 8)
20 FOR I = 1 TO 8
30 FOR J = 1 TO 8
→ 40 PRINT AT 2 * I, 2 * J + 8; "gH"
50 NEXT J
60 NEXT I
70 LET P = 2
80 LET Q = 9
90 GOSUB 2000
92 LET P0 = P
94 LET Q0 = Q
100 IF INKEY$ < > " " THEN GOTO 100
110 IF INKEY$ = " " THEN GOTO 110
120 LET I$ = INKEY$
130 IF I$ = "Q" THEN GOTO 200
140 IF I$ = "5" THEN LET Q = Q - 2
150 IF I$ = "6" THEN LET P = P + 2
160 IF I$ = "7" THEN LET P = P - 2
170 IF I$ = "8" THEN LET Q = Q + 2
180 GOSUB 1000
185 GOSUB 1020
190 GOTO 92
→ 200 PRINT AT P, Q + 1; " Q "
```

```

205 LET I = P/2
207 LET J = (Q - 7)/2
210 LET B (I, J) = 1
220 FOR T = 1 TO 8
230 IF B (I, T) = 0 THEN LET B (I, T) = -1
240 IF B (T, J) = 0 THEN LET B (T, J) = -1
250 NEXT T
260 FOR F = -1 TO 1 STEP 2
270 FOR T = -8 TO 8
280 LET U = I + F * T
290 LET V = J + T
300 IF U < 1 OR U > 8 OR V < 1 OR V > 8 THEN GOTO 320
310 IF B (U, V) = 0 THEN LET B (U, V) = -1
320 NEXT T
330 NEXT F
340 GOSUB 2000
350 GOTO 92
→ 1000 PRINT AT P, Q; " > "
1010 RETURN
→ 1020 PRINT AT P0, Q0; "□"
1030 RETURN
2000 FOR I = 1 TO 8
2010 FOR J = 1 TO 8
2020 IF B (I, J) = -1 THEN PRINT AT 2 * I, 2 * J + 8; "+"
2030 NEXT J
2040 NEXT I
2050 RETURN

```



USING THE ZX81 VERSION

RUN. You get a board. Use the arrow keys to move a cursor around. If you hit "Q" it gives you a queen on the board, at that position, and also shows you which squares are attacked by it. Now add another queen . . . keep going until the squares are all attacked.

SPECTRUM VERSION

Change the following lines:

```
40 PRINT AT 2 * I, 2 * J + 8; "□"  
200 PRINT AT P, Q + 1; "gQ"  
1000 [delete this line]  
1020 [delete this line]
```

Add some colour:

```
1 PAPER 7: INK 0: CLS  
35 PAPER 5 + I + J - 2 * INT ( ( I + J ) / 2 )  
1000 PRINT PAPER 7; AT P, Q; ">"  
1020 PRINT PAPER 7; AT P0, Q0; "□"
```

Set up the queen with user-defined graphics:

```
3000 DATA 170, 170, 84, 56, 56, 124, 254, 0  
3010 FOR N = 0 TO 7  
3020 READ X  
3025 POKE USR "Q" + N, X  
3030 NEXT N
```

USING THE SPECTRUM VERSION

On first switching on, type GO TO 3000 to set up the graphics. Then RUN thereafter. Otherwise as for the ZX81.

What Royalty can do, so can the Clergy:

8 bishops blatant

This is just like QUEENS DOMINANT, but with bishops.

ZX81 VERSION

Start with the ZX81 version of QUEENS DOMINANT, and make the following changes.

```
130 IF I$ = "B" THEN GOTO 200
200 PRINT AT P, Q + 1; " B "
220-250 [delete these lines]
```

USING THE ZX81 VERSION

Just like QUEENS DOMINANT, except you hit "B" to print a bishop.

SPECTRUM VERSION

Change the following lines in the Spectrum version of QUEENS DOMINANT:

```
130 IF I$ = "B" THEN GOTO 200
200 PRINT AT P, Q + 1; "gB"
220-250 [delete these lines]
3000 DATA 16, 24, 40, 52, 60, 60, 126, 0
3025 POKE USR "B" + N, X
```

USING THE SPECTRUM VERSION

GO TO 3000, on first switching on, to set up the graphics. After that it's just like (surprise) QUEENS DOMINANT, except that you use "B" to print a bishop.

. . . and so can the Military!

9 rooks rampant

Just like QUEENS DOMINANT (QD) and BISHOPS BLATANT (BB)—but with rooks.

ZX81 VERSION

Start from QUEENS DOMINANT, ZX81 version, and change:

```
130 IF I$ = "R" THEN GOTO 200
200 PRINT AT P, Q + 1; " R"
260-330 [delete these lines]
```

USING THE ZX81 VERSION

As for QD and BB, but hit "R" for rook.

SPECTRUM VERSION

Change QUEENS DOMINANT, Spectrum version, lines:

```
130 IF I$ = "R" THEN GOTO 200
200 PRINT AT P, Q + 1; "gR"
260-330 [delete these lines]
3000 DATA 84, 56, 56, 56, 56, 56, 124, 0
3025 POKE USR "R" + N, X
```

USING THE SPECTRUM VERSION

On first switch-on, GOTO 3000 to set up the rook graphics. After that, RUN. Use "R" to print a rook.

You'll need eight of them.

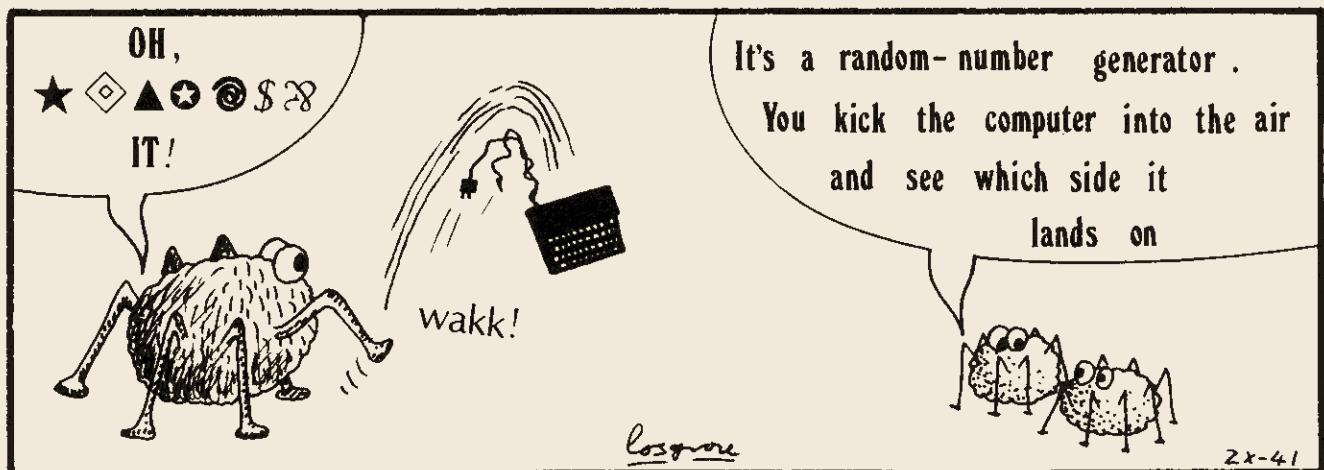
*I promise, this is the last
variation on VICARS VACANT . . .
or whatever the blasted thing
was . . .*

10 shy ladies

Not only that: there's no extra work! Same program as QUEENS DOMINANT; but new question: How many queens can you get on the board without any of them attacking any of the others?

Can you get eight on?

There are, of course, the Cautious Clergymen and the Reluctant Rooks . . . dammit, this rip-off has gone on long enough: let's see something new . . .



*A problem in division, to
drive you nuts . . .*

|| monkey puzzle

Five astronauts are cast away on a desert asteroid (well, this *is* 1982) whose sole inhabitant is an (alien) monkey called Knees. (Ever heard of Ape Knees? You've probably got some in your pocket . . .) The only food on the asteroid is a plentiful supply of coconuts.

The astronauts pick a quantity of coconuts (no, that can't be right, never mind . . .) and leave them in a heap.

During the night, the first astronaut wakes up and decides to nick his half of the coconuts (fair shares, right?). However, they won't divide exactly in two: there's one left over. Having left the nut-slicer at a sleazy astroway cafe halfway to Betelgeuse, he solves the problem by giving the spare one to the monkey. He takes his coconuts away and eats them, leaving the rest in the heap.

Soon after, astronaut 2 wakes up, and does exactly the same. Again, there is one nut left for the monkey after halving the pile.

Astronauts 3 and 4 in turn take their halves of the nuts. The monkey gets lucky each time; but astronaut 5 is a little perplexed, because only one nut is left for him. He vaguely thinks there were a few more than that when he went to bed the night before; but it *is* twenty-nine o'clock in the early morn, so his brain isn't functioning too well.

How many coconuts were there to start with?

ZX81 VERSION

```
→ 10 PRINT "HOW MANY COCONUTS"  
20 INPUT C  
30 GOSUB 500  
40 FOR T = 1 TO 5  
50 IF C = 1 THEN GOTO 700  
60 LET C = C - 1  
70 LET K = INT (C/2)  
80 IF 2 * K < > C THEN GOTO 600  
90 LET C = K  
100 GOSUB 500  
110 NEXT T
```

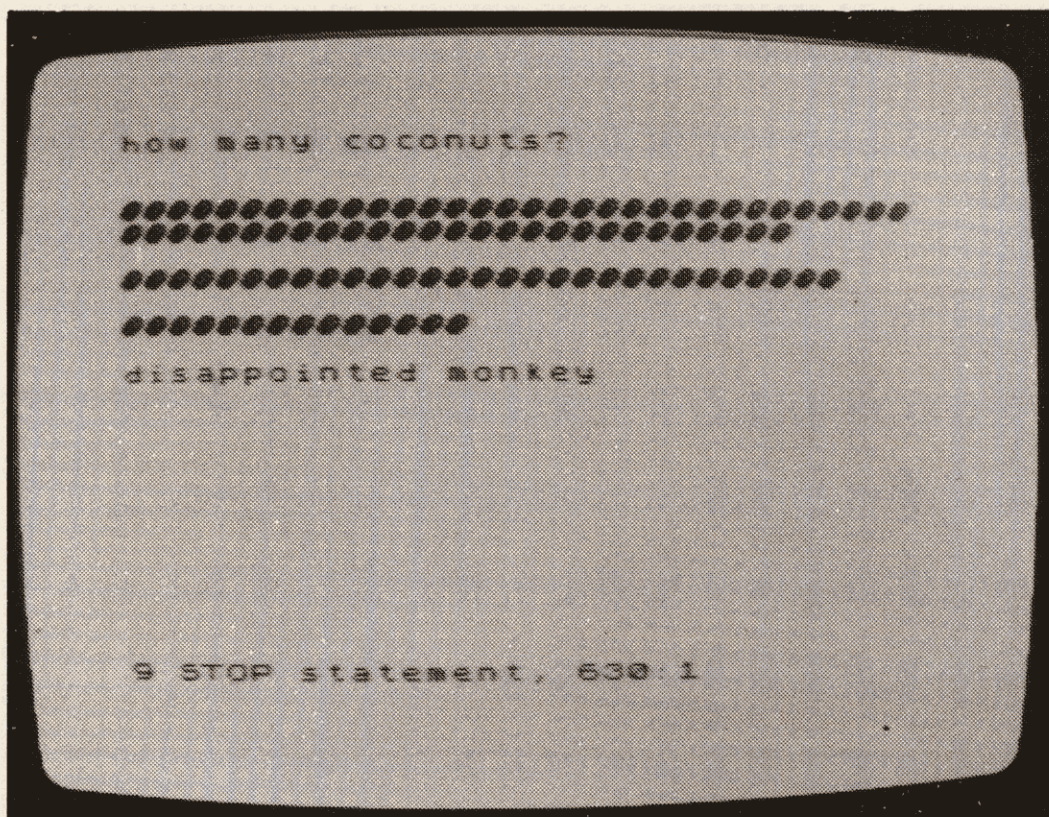
```

120 PRINT
130 PRINT
140 PRINT "TOO MANY"
150 STOP
500 PRINT
510 PRINT
520 FOR I = 1 TO C
530 PRINT "*";
540 NEXT I
550 RETURN
600 PRINT
610 PRINT
620 PRINT "DISAPPOINTED MONKEY"
630 STOP
700 PRINT
710 PRINT
→ 720 IF T = 5 THEN PRINT "THAT " " S RIGHT"
730 IF T < > 5 THEN PRINT "NOPE, TOO SMALL"

```

USING THE ZX81 VERSION

RUN. Now you can guess the number of coconuts. The computer runs through the night's events, printing out piles of coconuts, and checking whether everyone's getting what they should. If you're right, it says so. If wrong, ditto, and you get a clue for improvements.



SPECTRUM VERSION

Change these:

```
10 PRINT "How many coconuts?"
720 IF T = 5 THEN PRINT FLASH 1; "THAT " " S RIGHT!!!!!";
GOTO 800
```

Add these:

```
740 STOP
800 FOR I = 1 TO 50
810 BEEP .1, 40 * (.5 - RND); BORDER 7 * RND
820 NEXT I
```

Also, we can do a hi-res coconut; change this:

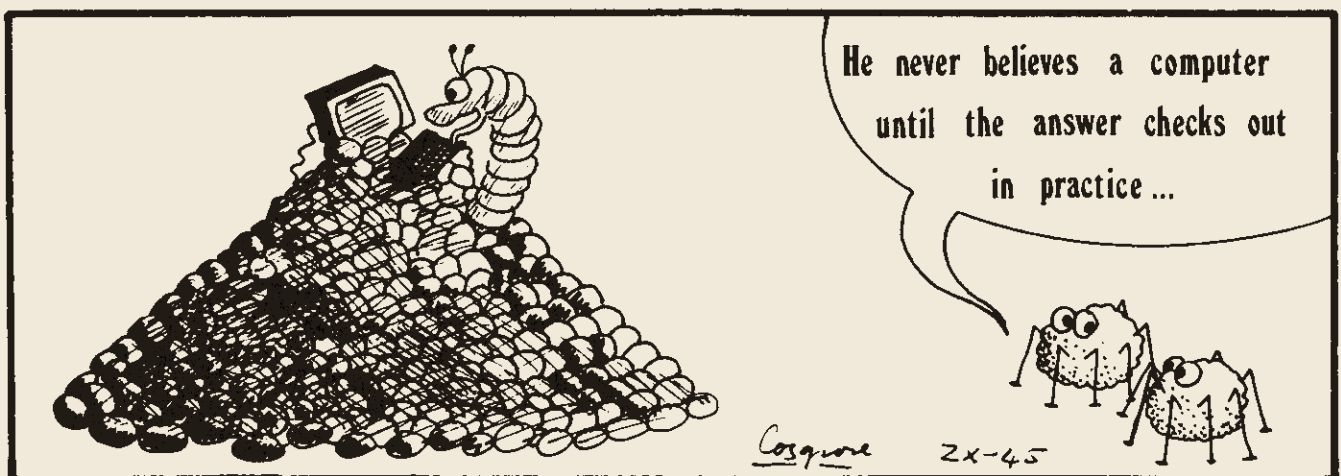
```
530 PRINT "gN";
```

And add these:

```
1000 DATA 28, 54, 122, 118, 252, 248, 112, 0
1010 FOR N = 0 TO 7
1020 READ X
1030 POKE USR "N" + N, X
1040 NEXT N
```

USING THE SPECTRUM VERSION

On first warm-up, GO TO 1000 for the coconuts; then RUN thereafter. Apart from that, follow the ZX81 instructions.



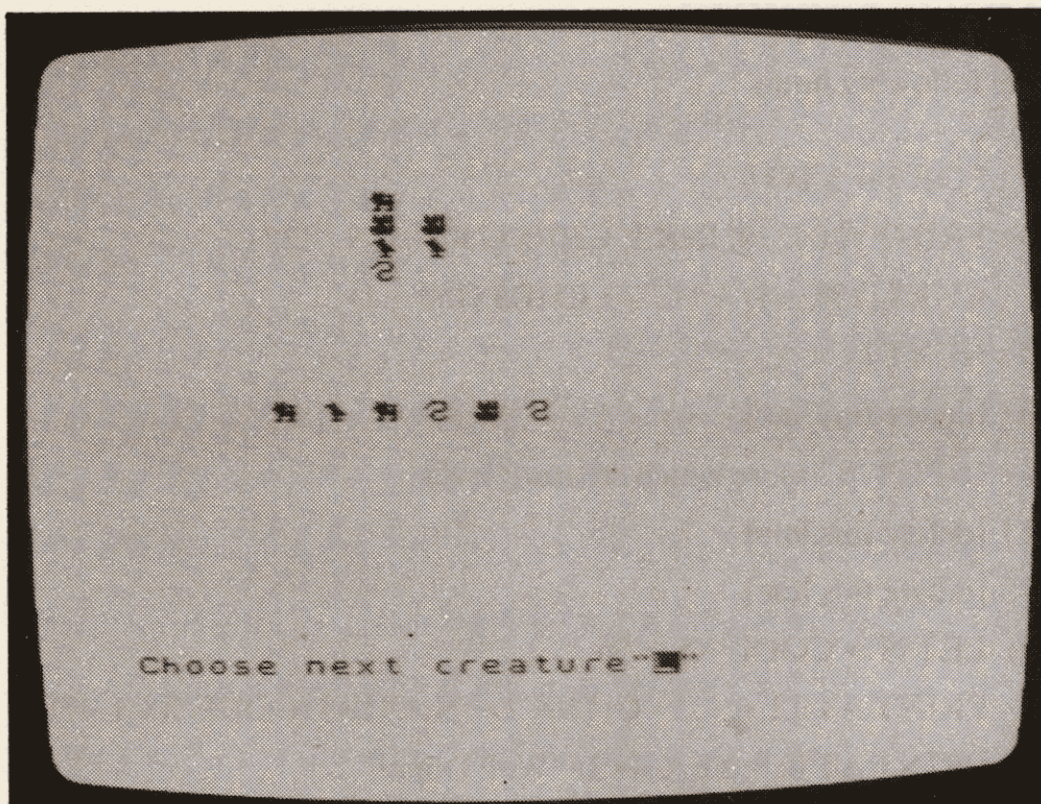
Arrange the animals amicably:

12 ecological chain

You've got three dogs, three cats, three birds, and three worms. Arrange them in a row so that no dog is next to a cat, no cat to a bird, and no bird to a worm (for obvious reasons, they get hungry and so forth)—*and* no two cats are together, no two dogs, no two birds, and no two worms. Even a worm has his pride.

ZX81 VERSION

```
10 DIM K (4)
→ 20 PRINT AT 3, 10; "D □ D □ D"
→ 30 PRINT AT 4, 10; "C □ C □ C"
→ 40 PRINT AT 5, 10; "B □ B □ B"
→ 50 PRINT AT 6, 10; "W □ W □ W"
60 LET C = 0
70 LET Y$ = "G"
```



```

→ 100 PRINT AT 0, 0; "CHOOSE WHICH CREATURE"
→ 102 INPUT X$
→ 104 PRINT AT 0, 0; " [21 spaces] "
→ 110 LET N = CODE X$ - 37 - 22 * (X$ = "W")
120 LET K (N) = K (N) + 1
130 IF K (N) > 3 THEN GOTO 500
→ 140 PRINT AT 12, 6 + 2 * C; X$
150 LET C = C + 1
160 PRINT AT 7 - N, 16 - 2 * K (N); "□"
180 IF ABS (CODE X$ - CODE Y$) < 2 THEN GOTO 600
190 LET Y$ = X$
200 IF C = 12 THEN GOTO 400
210 GOTO 100
→ 400 PRINT AT 0, 8; "WELL DONE"
420 STOP
500 PRINT AT 0, 8; "ILLEGAL MOVE"
510 PAUSE 50
520 PRINT AT 0, 8; " [12 spaces] "
530 GOTO 100
600 PRINT AT 0, 8; "AAAAAAAAAAGGGGGGGGHHHHH"

```

USING THE ZX81 VERSION

Type RUN. The available animals are listed by initial. Choose which one to start the row with by typing in its initial. It will be displayed. Continue: watch the messages.

SPECTRUM VERSION

Change the following lines:

```

20 FOR I = 3 TO 6
30 PRINT AT I, 10; INK I; CHR$ (150 - I) + "□" +
  CHR$ (150 - I) + "□" + CHR$ (150 - I)
40 NEXT I
50 [delete this line]
100 INPUT "Choose which creature"; X$
102 [delete this line]
104 [delete this line]
110 LET N = CODE X$ - 96
140 PRINT AT 12, 6 + 2 * C; INK 7 - N; CHR$ (CODE X$ + 47)
400 PRINT AT 0, 8; FLASH 1; "Well Done!"

```


Add these:

```
410 BEEP .3, 0: BEEP .3, 4: BEEP .3, 7: BEEP 1.5, 10
610 FOR I = 1 TO 10
620 BEEP I/15, - I
630 NEXT I
```

Set up hi-res graphics characters for the animals:

```
1000 DATA 60, 66, 130, 28, 32, 34, 28, 0, 16, 112, 56, 28,
      30, 8, 28, 0, 74, 122, 122, 50, 250, 254, 124,
      0, 80, 118, 242, 126, 62, 34, 102, 0
1010 FOR T = 1 TO 4
1020 FOR N = 0 TO 7
1030 READ X
1040 POKE USR CHR$(T + 143) + N, X
1050 NEXT N
1060 NEXT T
```

USING THE SPECTRUM VERSION

On first switch-on, GOTO 1000 for the graphics to be set up. After that, RUN. When asked for the animal, input

A for a worm (NB, *not* W!)
B for a bird
C for a cat
D for a dog.

Otherwise, it's like the ZX81 version.

*This is an oldie, but it's great fun,
and the graphics are nice . . .*

13 tower of Hanoi with pancakes

You have three plates. On one is a stack of pancakes of sizes 1, 2, 3, 4, . . . piled with the larger ones underneath. You have to transfer the whole stack to another plate, by:

- (a) moving only one pancake at a time;
- (b) only placing it on an empty plate, or on top of a *larger* pancake.

These rules aside, you can make as many moves as you want to.

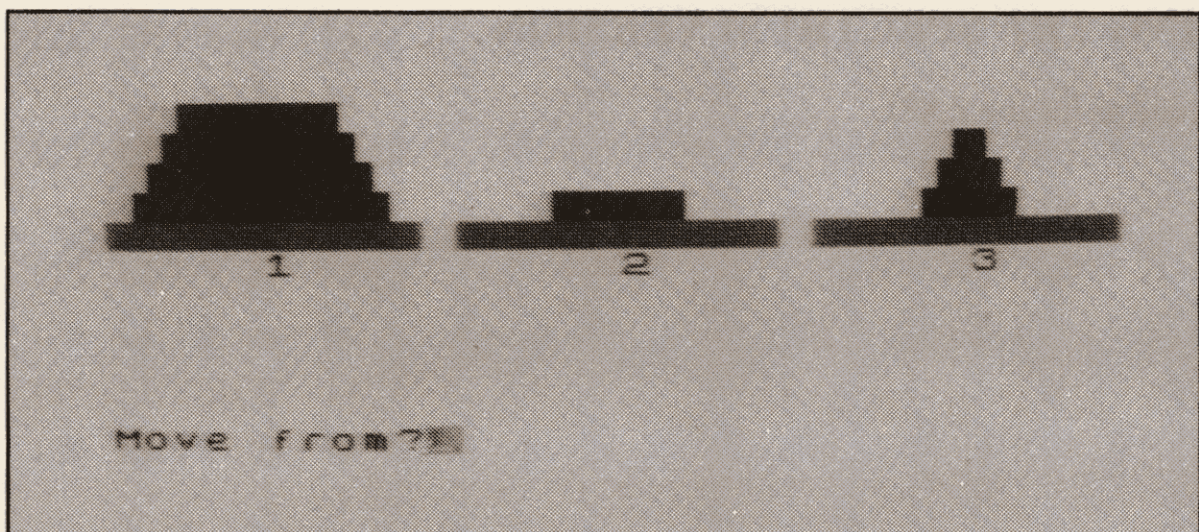
ZX81 VERSION

```
10 LET E$ = "■■■■■■■■■■"
15 LET F$ = "□□□□□□□□□□"
→ 18 LET G$ = "gH gH gH gH gH gH gH gH gH"
→ 20 PRINT AT 0, 0; "INPUT SIZE"
→ 22 INPUT S
→ 24 PRINT AT 0, 0; "[10 spaces]"
25 LET V = INT ((11 - S) / 2)
30 DIM D$(S, S + 2)
40 FOR T = S TO 1 STEP - 2
50 LET D$(T) = F$(TO (S - T) / 2) + E$(TO T)
→ 60 IF T < S THEN LET D$(T + 1) = F$(TO (S - T) / 2 - 1) + "g8" +
    E$(TO T) + "g5"
70 NEXT T
→ 80 IF S = 2 * INT (S/2) THEN LET D$(1) =
    F$(TO (S - 2) / 2) + "g8 g5"
→ 100 PRINT AT 16, 0; G$ + "□" + G$ + "□" + G$
110 FOR I = 1 TO 3
120 PRINT AT 17, 11 * I - 6; I
130 NEXT I
200 FOR I = 1 TO S
210 PRINT AT 15 - S + I, V; D$(I)
```

```

220 NEXT I
300 DIM A (3, S)
310 FOR I = 1 TO S
320 LET A (1, I) = S - I + 1
330 NEXT I
340 DIM H (3)
350 LET H (1) = S
→ 500 PRINT AT 0, 0; "MOVE FROM"
→ 502 INPUT F
→ 504 PRINT AT 0, 0; "MOVE TO □ □"
→ 506 INPUT G
→ 508 PRINT AT 0, 0; "[9 spaces]"
511 GOTO 1000
512 LET R = A (F, H (F) )
514 LET A (F, H (F) ) = 0
516 LET A (G, H (G) + 1) = R
520 LET H (F) = H (F) - 1
530 LET H (G) = H (G) + 1
540 LET X = R + 1 + .5 * (10 - R)
545 IF X > S + 2 THEN LET X = S + 2
550 PRINT AT 15 - H (F), 11 * F - 11 + V; F$ (TO X);
    AT 16 - H (G), 11 * G - 11 + V; D$ (R) (TO X)
→ 555 PRINT AT 16, 0; G$
557 IF H (2) = S OR H (3) = S THEN GOTO 1200
560 GOTO 500
1000 IF H (G) = 0 THEN GOTO 512
1005 IF H (F) = 0 THEN GOTO 500
1010 IF A (F, H (F) ) >= A (G, H (G) ) THEN GOTO 500
1020 GOTO 512
→ 1200 PRINT AT 1, 10; "SUCCESS"

```



USING THE ZX81 VERSION

Start with RUN; wait for the tower to be drawn. Choose which pile to move from, and which to, by inputting their numbers. Keep going until you transfer them all to another plate.

SPECTRUM VERSION

Change the following lines:

```
18  [delete this line]
20  INPUT "Size?"; S
22  [delete this line]
24  [delete this line]
60  IF T < S THEN LET D$(T + 1) = F$(TO (S - T) / 2 - 1) + "g5" +
    E$(TO T) + "g5c"
80  IF S = 2 * INT (S/2) THEN LET D$(1) =
    F$(TO (S - 2) / 2) + "g5 g5c"
100 PRINT AT 16, 0; INK 4; E$ + "□" + E$ + "□" + E$
500 INPUT "Move from?"; F
502 [delete this line]
504 INPUT "Move to?"; G
506 [delete this line]
508 [delete this line]
555 PRINT AT 16, 0; INK 4; E$
1200 PRINT AT 1, 10; FLASH 1; "Success!"
```

Now add these:

```
1210 FOR I = 1 TO 80
1220 BORDER 7 * RND
1230 BEEP 2/(20 + I), I/2 + 10 * (RND - .5)
1240 NEXT I
```

USING THE SPECTRUM VERSION

Proceed as for the ZX81.

Swap the animals using
knight's moves:

19 several times knightly

To start with, two species of animal are arranged at the corners of a 3×3 square, like this:

```
A □ A
□   □
B □ B
```

Your job is to swap the A's and B's. Any animal can hop into an *empty* space, by a move like that of a knight in *chess*. (Two steps sideways and one at right angles to that.)

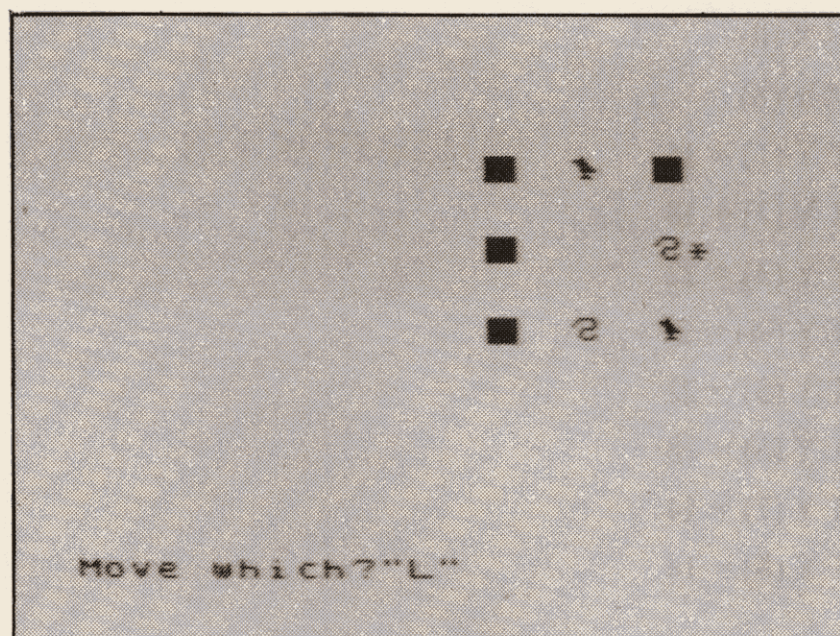
ZX81 VERSION

```
10 DIM A (3, 3)
20 LET A (1, 1) = 1
30 LET A (1, 3) = 1
40 LET A (3, 1) = 2
50 LET A (3, 3) = 2
60 LET A (2, 2) = 3
70 LET C = 1
100 DIM X (8)
110 DIM Y (8)
121 LET Y (1) = 14
122 LET Y (2) = 18
123 LET Y (3) = 22
124 LET Y (4) = 22
125 LET Y (5) = 22
126 LET Y (6) = 18
127 LET Y (7) = 14
128 LET Y (8) = 14
```

```

131 LET X (1) = 7
132 LET X (2) = 7
133 LET X (3) = 7
134 LET X (4) = 11
135 LET X (5) = 15
136 LET X (6) = 15
137 LET X (7) = 15
138 LET X (8) = 11
200 GOSUB 1000
210 PRINT AT X (1), Y (1); "*"
→ 700 PRINT AT 0, 0; "MOVE WHICH"
→ 702 INPUT I$
710 IF I$ < > " " THEN GOTO 719
715 GOSUB 1100
718 GOTO 700
719 LET C0 = C
→ 720 PRINT AT 0, 0; "WHERE □ □ □ □ □"
→ 722 INPUT I$
→ 725 PRINT AT 0, 0; " [10 spaces] "
730 IF I$ < > " " THEN GOTO 750
735 GOSUB 1100
740 GOTO 720
750 LET C1 = C
760 IF ABS (C0 - C1) = 3 OR ABS (C0 - C1) = 5 THEN GOTO 2000
770 PRINT AT 0, 0; "ILLEGAL"
780 PAUSE 50
790 PRINT AT 0, 0; " [7 spaces] "
800 GOTO 700

```



```

1000 FOR I = 1 TO 3
1010 FOR J = 1 TO 3
1020 IF A (I, J) = 0 THEN LET Q$ = "■"
→ 1030 IF A (I, J) = 1 THEN LET Q$ = "A"
→ 1040 IF A (I, J) = 2 THEN LET Q$ = "B"
1050 IF A (I, J) = 3 THEN LET Q$ = "□"
1060 PRINT AT 5 + 3 * I, 12 + 3 * J; Q$
1070 NEXT J
1080 NEXT I
1090 RETURN
1100 PRINT AT X (C), Y (C); "□"
1110 LET C = C + 1
1120 IF C > 8 THEN LET C = 1
1130 PRINT AT X (C), Y (C); "*"
1140 RETURN
2000 LET X 0 = (X (C0) - 3) / 4
2010 LET Y0 = (Y (C0) - 10) / 4
2020 LET X1 = (X (C1) - 3) / 4
2030 LET Y1 = (Y (C1) - 10) / 4
2040 LET R = A (X0, Y0)
2045 IF R = 0 THEN GOTO 770
2050 LET A (X0, Y0) = A (X1, Y1)
2060 LET A (X1, Y1) = R
2070 GOSUB 1000
2080 IF A (1, 1) = 2 AND A (1, 3) = 2 AND
    A (3, 1) = 1 AND A (3, 3) = 1 THEN GOTO 3000
2100 GOTO 700
→ 3000 PRINT AT 0, 8; "GOTCHA"

```

USING THE ZX81 VERSION

RUN. The board will be displayed, with the animals marked A and B. An asterisk acts as cursor. If you press "NEWLINE" it moves clockwise; this repeats until you input any non-empty string. Then the cursor position defines which animal will move. You select where it moves to using the cursor the same way. The move is then made, and you may continue.

SPECTRUM VERSION

Change these lines:

```
700 INPUT "Move which?"; I$
702 [delete this line]
720 INPUT "Where?"; I$
722 [delete this line]
725 [delete this line]
1030 IF A (I, J) = 1 THEN LET Q$ = "gA"
1040 IF A (I, J) = 2 THEN LET Q$ = "gB"
3000 PRINT AT 0, 8; FLASH 1; "Gotcha!!!!"
```

And add these:

```
3010 FOR I = 1 TO 80
3020 BORDER 7 * RND
3030 BEEP 1/(10 + I), I/2 + 7 * (RND - .3)
3040 NEXT I
```

Set up the graphics characters for "gA" and "gB" to your favourite animals. You could use the cat and dog from ECOLOGICAL CHAIN: there they are POKEd into "C" and "D" graphics characters, but you can get them into "A" and "B" as follows:

```
4000 DATA 74, 122, 122, 50, 250, 254, 124, 0,
80, 118, 242, 126, 62, 34, 102, 0
4010 FOR N = 0 TO 7
4020 READ X
4030 POKE USR "A" + N, X
4040 NEXT N
4050 FOR N = 0 TO 7
4060 READ X
4070 POKE USR "B" + N, X
4080 NEXT N
```

USING THE SPECTRUM VERSION

GOTO 4000 on first switching on; then use RUN thereafter. It works just like the ZX81 version, but it's prettier.

And, finally, another animal swap-shop:

15 beasts on the steps

Five cats and five dogs sit on a flight of 12 steps. You can move any adjacent pair into the blank space. Repeat this until you get the five cats at the top, then the five dogs, and the last two steps are blank.

ZX81 VERSION

```
10 LET A$ = " [32 inverse spaces] "
→ 15 LET B$ = "ABABABAB □ □"
20 PRINT AT 5, 0
30 FOR I = 1 TO 12
40 PRINT A$ (TO 15 + I)
50 NEXT I
60 PRINT A$ + A$ + A$ + A$
70 FOR I = 1 TO 12
80 PRINT AT 5 + I, 12 + I + (1 < 10); I
90 NEXT I
95 LET P = 11
100 FOR I = 1 TO 5
→ 110 PRINT AT 4 + 2 * I, 14 + 2 * I; "A"
→ 120 PRINT AT 5 + 2 * I, 15 + 2 * I; "B"
130 NEXT I
→ 200 PRINT AT 0, 0; "MOVE WHICH"
→ 201 INPUT N
202 IF N < 1 OR N > 11 THEN GOTO 200
→ 203 PRINT AT 0, 0; " [10 spaces] "
205 LET X$ = B$ (N)
206 LET Y$ = B$ (N + 1)
208 IF X$ = "□" OR Y$ = "□" THEN GOTO 500
```

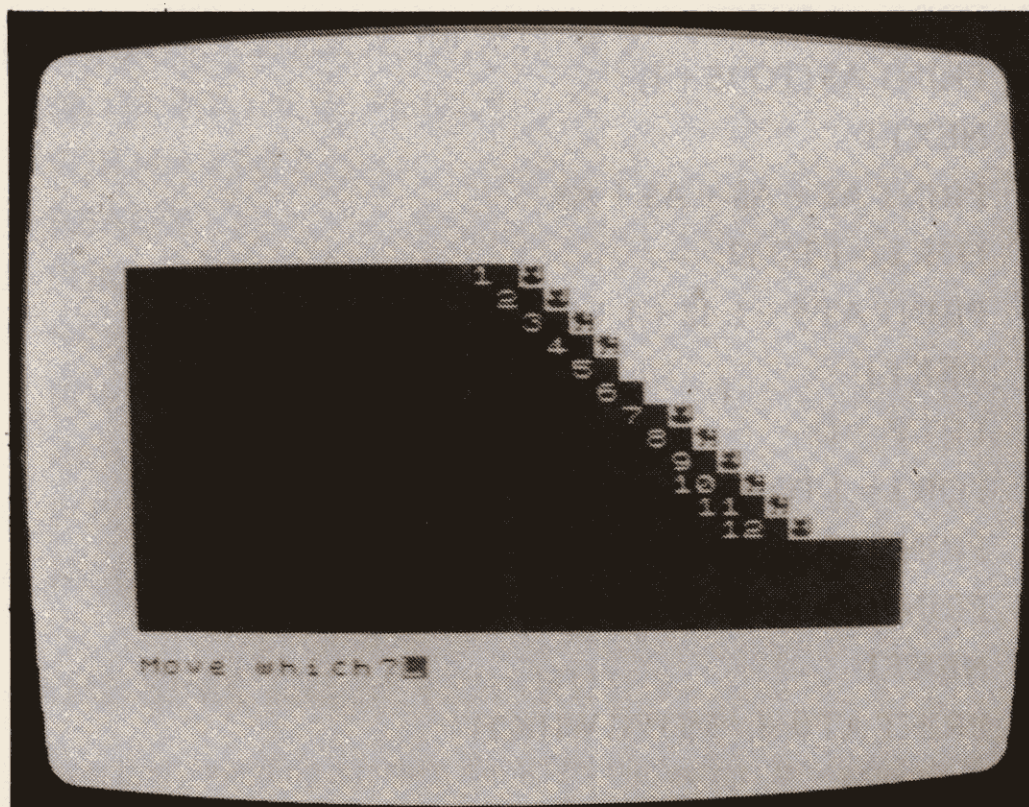
```

210 PRINT AT 5 + N, 15 + N; "□"; AT 6 + N, 16 + N; "□";
    AT 5 + P, 15 + P; X$; AT 6 + P, 16 + P; Y$
211 LET B$(N) = "□"
212 LET B$(N + 1) = "□"
213 LET B$(P) = X$
214 LET B$(P + 1) = Y$
220 LET P = N
→ 225 IF B$ = "AAAAABBBBB □ □" THEN GOTO 1000
230 GOTO 200
500 PRINT AT 0, 0; "ILLEGAL"
510 PAUSE 50
520 PRINT AT 0, 0; " [7 spaces] "
530 GOTO 200
→ 1000 PRINT AT 1, 8; "DONE IT"

```

USING THE ZX81 VERSION

RUN. You specify the *first* of the two adjacent animals you wish to move, by the number of the corresponding step. All else is automatic.



SPECTRUM VERSION

Change these lines:

```
15 LET B$ = "gA gB gA gB gA gB gA gB gA gB □ □"  
110 PRINT AT 4 + 2 * I, 14 + 2 * I; "gA"  
120 PRINT AT 5 + 2 * I, 15 + 2 * I; "gB"  
200 INPUT "Move which?"; N  
201 [delete this line]  
203 [delete this line]  
225 IF B$ = "gA gA gA gA gA gB gB gB gB □ □"  
    THEN GOTO 1000  
1000 PRINT AT 1, 8; FLASH 1; "DONE IT!"
```

Add these lines:

```
1020 FOR I = 1 TO 30  
1030 BORDER 4: BEEP .1, 12: BORDER 2: BEEP .1, 0  
1040 NEXT I
```

Set up the user-defined graphics:

```
2000 DATA 37, 61, 61, 25, 125, 127, 62, 0,  
    40, 59, 121, 63, 31, 17, 51, 0  
2010 FOR N = 0 TO 7  
2020 READ X  
2030 POKE USR "A" + N, X  
2040 NEXT N  
2050 FOR N = 0 TO 7  
2060 READ X  
2070 POKE USR "B" + N, X  
2080 NEXT N
```

(These numbers are exact halves of those used in SEVERAL TIMES KNIGHTLY. If you wonder why, *don't* halve them, and see . . .)

USING THE SPECTRUM VERSION

On first switch-on, RUN ~~2000~~ or GOTO ~~2000~~. Thereafter, RUN is fine. Apart from that, the instructions are as for the ZX81 version.

WARNING!!

spectrumizers only past this point!

Here are a few programs which will only run on the ZX Spectrum. Well, we thought you ought to be given some justification for having bought the all-singing, all-dancing version, rather than the steam-driven ZX81. And it might give all you ZX81ers an excuse to rush out and buy Spectrums. And that means we'll sell lots more copies of our book *Easy Programming for the ZX Spectrum*.

Well, we can dream can't we?

*Here's an old favourite with some twists
you may not have seen before*

16 solitaire

This is the standard game of solitaire. The pegs are displayed as asterisks, and the four corners, which cannot be played in, show up as capital O's. The rows are labelled a-g on the screen, and the columns are labelled 1-7.

The computer prompts you for a move in the order "square to be moved from", "square to be moved to". Enter these in the order letter, digit. So you could start by moving f4 into the centre square, d4.

You would type:

f4 (Enter) d4 (Enter)

and the computer will move the pegs for you. Watch it! It notices if you cheat!

Just in case you're not familiar with this puzzle here's a quick resumé of the rules:

1. The object of the game is to remove all but one of the pegs from the board.
2. A legal move consists of taking a peg and leap-frogging an adjacent peg (vertically or horizontally, but not diagonally) into an empty square. The leap-frogged peg is removed from the board.

```
10 LET n = 32
20 LET getmove = 1000: LET testmove = 2000
30 LET makemove = 3000
40 FOR r = 4 TO 16 STEP 2
50 FOR c = 9 TO 21 STEP 2
60 PRINT PAPER 5: AT r, c: "O"
70 NEXT c
80 NEXT r
100 FOR c = 13 TO 17 STEP 2
110 FOR r = 4 TO 16 STEP 2
120 PRINT PAPER 6: AT r, c: "*"
130 NEXT r
140 NEXT c
150 FOR r = 8 TO 12 STEP 2
160 FOR c = 9 TO 21 STEP 2
170 PRINT PAPER 6: AT r, c: "*"
180 NEXT c
190 NEXT r
```

```

200 PRINT PAPER 6; OVER 1; AT 10, 15; "*"
210 FOR r = 4 TO 16 STEP 2
220 PRINT AT r, 7; CHR$(r/2 + 95)
230 NEXT r
240 FOR c = 9 TO 21 STEP 2
250 PRINT AT 18, c; CHR$(c/2 + 44)
260 NEXT c
270 GO SUB getmove
280 GO SUB testmove
290 IF NOT legal THEN PRINT AT 0, 0; "Illegal Move!!": GO TO 270
300 PRINT AT 0, 0; "□□□□□□□□□□□□□□" [14 spaces]
310 GO SUB makemove: LET n = n - 1
320 IF n > 1 THEN GO TO 270
330 PRINT AT 0, 0; "You've done it!"
340 GO TO 9999

1000 INPUT "from to: "; f$, t$
1010 LET rf = 2 * (CODE f$ (1) - 95)
1020 LET cf = 2 * (CODE f$ (2) - 45) + 1
1030 LET rt = 2 * (CODE t$ (1) - 95)
1040 LET ct = 2 * (CODE t$ (2) - 45) + 1
1050 RETURN

2000 LET legal = 1
2010 IF SCREEN$(rf, cf) <> "*" THEN LET legal = 0
2020 IF SCREEN$(rt, ct) <> "□" THEN LET legal = 0
2025 IF SCREEN$( (rf + rt) / 2, (cf + ct) / 2) <> "*"
    THEN LET legal = 0
2030 IF ABS (rf - rt) = 4 AND cf <> ct THEN LET legal = 0
2040 IF ABS (cf - ct) = 4 AND rf <> rt THEN LET legal = 0
2050 IF ABS (rf - rt) <> 4 AND rf <> rt THEN LET legal = 0
2060 IF ABS (cf - ct) <> 4 AND cf <> ct THEN LET legal = 0
2070 RETURN

3000 PRINT PAPER 6; OVER 1; AT rf, cf; "*"
3010 PRINT PAPER 6; AT rt, ct; "*"
3020 PRINT PAPER 6; OVER 1; AT (rf + rt) / 2, (cf + ct) / 2; "*"
3030 RETURN

```

There are other versions of the game, in which the rules for peg removal are just the same, but the number of pegs, and their starting positions differ.

One of the simplest is the “Latin Cross”. You can generate this shape by editing lines 10, 120 and 170 as follows:

```
10 LET n = 6
120 PRINT PAPER 6; AT r, c; “□”
170 PRINT PAPER 6; AT r, c; “□”
```

Now replace line 200:

```
200 GO SUB 4000
```

and add this routine:

```
4000 FOR r = 6 TO 12 STEP 2
4010 PRINT PAPER 6; AT r, 15; “*”
4020 NEXT r
4030 PRINT PAPER 6; AT 8, 13; “*”
4040 PRINT AT 8, 17; “*”
4050 RETURN
```

To form a “Greek Cross”, make exactly the same modifications as for the Latin Cross, but replace the whole subroutine from line 4000 onwards with:

```
4000 FOR r = 6 TO 14 STEP 2
4010 PRINT AT r, 15; “*”
4020 NEXT r
4030 FOR c = 11 TO 19 STEP 2
4040 PRINT PAPER 6; AT 8, 17; “*”
4050 NEXT c
4060 RETURN
```

And edit line 10:

```
10 LET n = 9
```

Finally here’s one known as the “tilted square”. You’ll see why when it’s drawn!
Take the *original* program and add the following lines:

```
201 PRINT PAPER 6; OVER 1; AT 4, 13; “*”
202 PRINT PAPER 6; OVER 1; AT 4, 17; “*”
203 PRINT PAPER 6; OVER 1; AT 8, 21; “*”
204 PRINT PAPER 6; OVER 1; AT 12, 21; “*”
205 PRINT PAPER 6; OVER 1; AT 16, 17; “*”
206 PRINT PAPER 6; OVER 1; AT 16, 13; “*”
207 PRINT PAPER 6; OVER 1; AT 12, 9; “*”
208 PRINT PAPER 6; OVER 1; AT 8, 9; “*”
```

Then edit line 10:

```
10 LET n = 24
```


How quickly can you unjumble letters?

17 anagrams against the clock

The computer displays an anagram of a ten-letter word, and then starts to count down from 999 to 0. It takes just under 50 seconds to get there. You have to solve the anagram inside this time. As soon as you enter the first letter, the clock will stop, but don't try to use this feature for increasing your thinking time because you only get one chance and you can't delete a letter once it's been entered!

- 1 LET cont = 350: LET end = 370
- 2 LET delay = 30
- 3 LET pause = 1000
- 4 LET begin = 90
- 5 LET clear = 2000
- 10 DATA "disability"
- 11 DATA "indefinite"
- 12 DATA "perfidious"
- 13 DATA "improbable"
- 14 DATA "practising"
- 15 DATA "caretakers"
- 16 DATA "medicament"
- 17 DATA "reciprocal"
- 18 DATA "procedures"
- 19 DATA "elementary"
- 20 DATA "regardless"
- 21 DATA "algorithms"
- 22 DATA "traversing"
- 23 DATA "represents"
- 24 DATA "preciously"
- 25 DATA "completing"
- 26 DATA "subroutine"
- 27 DATA "inversions"
- 28 DATA "cryptogram"
- 29 DATA "statements"

```

30 DIM w$ (10)
90 RANDOMIZE
100 RESTORE (INT (RND * 20) + 10)
110 READ w$
115 LET x$ = w$
120 FOR i = 1 TO 20
130 LET p = INT (RND * 10) + 1
140 LET q = INT (RND * 10) + 1
150 LET t$ = w$ (p)
160 LET w$ (p) = w$ (q)
170 LET w$ (q) = t$
180 NEXT i
190 PRINT AT 2, 2; "Can you solve this anagram?"
200 PRINT AT 6, 10; w$
210 PRINT AT 10, 2; "Your time starts now"
215 PRINT AT 17, 10; "-----"
220 FOR i = 999 TO 0 STEP -1
225 IF i > 99 THEN PRINT AT 10, 25; i: GO TO 230
226 IF i > 9 THEN PRINT AT 10, 25; "□"; i: GO TO 230
227 PRINT AT 10, 25; "□ □"; i
230 LET c$ = INKEY$
240 IF c$ = " " THEN GO TO cont
250 LET w$ (1) = c$
260 PRINT AT 16, 10; w$ (1)
265 GO SUB pause
270 FOR p = 2 TO 10
280 LET c$ = INKEY$
290 IF c$ = " " THEN GO TO 280
300 LET w$ (p) = c$
305 PRINT AT 16, p + 9; c$
310 LET c$ = " "
315 GO SUB pause
320 NEXT p
330 IF w$ = x$ THEN PRINT AT 20, 2; "Well done!!!": GO TO end
340 PRINT AT 20, 2; "Wrong! □ □ The word was . . . . □ □ □ □ □ □ □";
FLASH 1; x$: GO TO end
350 NEXT i
360 PRINT AT 20, 2; "Out of time . . . The word was . . . □ □ □";
FLASH 1; x$

```


*Here's a favourite puzzle book game
given a little bit of Spectrum pzazz*

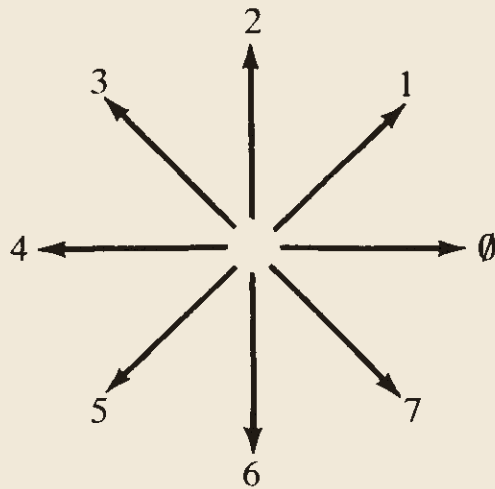
18 hidden words

Give the computer up to ten words, each up to ten letters long. (A ten-letter word will just go to the right-hand edge of the screen when you input it, so the rule is that you can't have a word which is split over two lines).

The computer will hide the words, horizontally, vertically or diagonally, backwards or forwards, in an otherwise random square of letters. When you find one, tell the computer where it starts. For instance, if it's in square g12 type:

g (Enter) 12 (Enter)

Then enter a number which indicates which direction the word takes from there. The direction values are:



Then say what the word is. If you're right, the computer will delete the word from the list of missing words, and colour it in, on the square.

If you enter a word which is on the square, but isn't in the list of missing words, the computer reckons that's cheating and refuses to play with you any more!

```
1 LET fillin = 1000: LET test = 2000
2 LET colour = 3000: LET rubout = 4000:
  LET list = 5000: LET delete = 6000
3 LET n = 0
4 RANDOMIZE
5 CLS
6 LET pc = 1
10 DIM w$(11, 11)
20 DIM s$(18, 18)
```

```

100 INPUT "How many words"; nw
110 FOR i = 1 TO nw
120 INPUT "Give me a word please"; w$ (i)
130 NEXT i
140 FOR i = 1 TO nw
150 LET d = INT (RND * 8)
160 LET r = INT (RND * 18) + 1
170 LET c = INT (RND * 18) + 1
180 GO SUB fillin
190 IF NOT filled THEN GO TO 150
200 NEXT i
210 FOR r = 1 TO 18
220 FOR c = 1 TO 18
230 IF s$ (r, c) = "□" THEN LET s$ (r, c) = CHR$ (INT (RND * 26) + 97)
240 NEXT c
250 NEXT r
260 FOR r = 1 TO 18
270 PRINT AT r, 0; CHR$ (r + 96)
280 PRINT AT r, 2; s$ (r)
290 NEXT r
300 PRINT AT 20, 2; "□□□□□□□□□□1111111111"
310 PRINT AT 21, 2; "123456789012345678"
315 GO SUB list
320 INPUT "Word start"; b$, bn
322 PRINT AT 16, 25; "□□□□□□"
325 INPUT "and direction"; d
330 INPUT "Which word is it"; f$
340 GO SUB test
350 IF valid THEN GO SUB colour
360 IF n < nw THEN GO TO 320
361 PRINT FLASH 1; AT 16, 22; "Well done"
370 GO TO 9999

1000 LET p = 1
1010 IF w$ (i, p) = "□" THEN LET filled = 1: RETURN
1020 IF r > 18 OR r < 1 OR c > 18 OR c < 1 THEN GO SUB rubout:
    LET filled = 0: RETURN
1030 IF s$ (r, c) < > "□" THEN GO SUB rubout: LET filled = 0: RETURN
1035 LET s$ (r, c) = w$ (i, p)
1040 LET p = p + 1

```

```

1050 GO TO 1100 + d * 10
1100 LET c = c + 1: GO TO 1010
1110 LET c = c + 1: LET r = r - 1: GO TO 1010
1120 LET r = r - 1: GO TO 1010
1130 LET c = c - 1: LET r = r - 1: GO TO 1010
1140 LET c = c - 1: GO TO 1010
1150 LET c = c - 1: LET r = r + 1: GO TO 1010
1160 LET r = r + 1: GO TO 1010
1170 LET r = r + 1: LET c = c + 1: GO TO 1010

2000 LET r = CODE b$ - 96
2010 LET c = bn
2020 FOR p = 1 TO LEN f$
2030 IF f$(p) < > s$(r, c) THEN PRINT FLASH 1; AT 16, 25; "Wrong!":
    LET valid = 0: RETURN
2040 GO TO 2100 + 10 * d
2100 LET c = c + 1: GO TO 2180
2110 LET c = c + 1: LET r = r - 1: GO TO 2180
2120 LET r = r - 1: GO TO 2180
2130 LET c = c - 1: LET r = r - 1: GO TO 2180
2140 LET c = c - 1: GO TO 2180
2150 LET c = c - 1: LET r = r + 1: GO TO 2180
2160 LET r = r + 1: GO TO 2180
2170 LET r = r + 1: LET c = c + 1: GO TO 2180
2180 NEXT p
2190 LET valid = 1
2195 LET n = n + 1
2197 GO SUB delete
2200 RETURN

3000 LET r = CODE b$ - 96
3005 LET pc = pc + 1
3006 IF pc > 6 THEN LET pc = 1
3010 LET c = bn + 1
3020 FOR p = 1 TO LEN f$
3030 PRINT PAPER pc; AT r, c; f$(p)
3040 GO TO 3100 + 10 * d
3100 LET c = c + 1: GO TO 3180
3110 LET c = c + 1: LET r = r - 1: GO TO 3180
3120 LET r = r - 1: GO TO 3180
3130 LET c = c - 1: LET r = r - 1: GO TO 3180

```

```

3140 LET c = c - 1: GO TO 3180
3150 LET c = c - 1: LET r = r + 1: GO TO 3180
3160 LET r = r + 1: GO TO 3180
3170 LET r = r + 1: LET c = c + 1: GO TO 3180
3180 NEXT p
3185 PAPER 7
3190 RETURN

4000 IF p = 1 THEN RETURN
4010 LET p = p - 1
4020 GO TO 4100 + 10 * d
4100 LET c = c - 1: GO TO 4180
4110 LET c = c - 1: LET r = r + 1: GO TO 4180
4120 LET r = r + 1: GO TO 4180
4130 LET c = c + 1: LET r = r + 1: GO TO 4180
4140 LET c = c + 1: GO TO 4180
4150 LET c = c + 1: LET r = r - 1: GO TO 4180
4160 LET r = r - 1: GO TO 4180
4170 LET r = r - 1: LET c = c - 1: GO TO 4180
4180 LET s$(r, c) = "□"
4190 GO TO 4000

5000 PRINT AT 0, 22; "Missing"
5010 PRINT AT 1, 22; "words . ."
5020 FOR i = 1 TO nw
5030 PRINT PAPER 6; AT i + 2, 21; w$(i)
5040 NEXT i
5050 RETURN

6000 LET w$(11) = f$
6005 FOR i = 1 TO nw
6010 IF w$(11) = w$(i) THEN PRINT AT i + 2, 21;
    "□□□□□□□□□□□": RETURN
6020 NEXT i

```

See how you would fare in Military Intelligence!

19 cryptograms

The computer will select a sentence, and display an encyphered version (without spaces between words—that would be a real giveaway!) on the screen. You have to work out the code, and key in the original sentence. You can put spaces in the decoded sentence to make it more readable. Careful though—you only get two guesses!

Don't think that when you've broken the code, the puzzle is finished. The computer changes its coding system every time it codes a new sentence!

Footnote (i) During the First World War, British Naval Intelligence invented a number of cyphering systems. To test one, which they were rather proud of, they encyphered a short message and gave it to their American counterparts to see if they could crack it. The following day, back came the clear text message:

“This code is absolutely unbreakable”

By a curious coincidence, this is 31 letters long (without spaces) so it just fits (with one spare position) on a line on the screen. So I've chosen it as one of the messages to decode. The other messages all have 31 letters too, so it's no good just counting how many there are in the coded message.

Footnote (ii) There's obviously a big clue to solving one of these cryptograms, because you know the answer has to be one of the five sentences in the DATA statements, so I'm not going to give you much help. But I will tell you that the cypher the computer chooses is always a simple substitution cypher, which means that if, for instance, “a” codes as “g” at the beginning of the message, it will still code “a” as “g” at the end. If you want to make things more difficult for yourself, why not get a friend to change the DATA statements? After that, no peeking!

```
1 RANDOMIZE
2 DIM w$(5, 31): DIM t(26): DIM n$(31)
3 DATA "this code is absolutely unbreakable"
4 DATA "the sentence in front of you is in code"
5 DATA "translating codes is a black art form"
6 DATA "cryptologists are men who study code"
7 DATA "a cryptographer deciphers messages"
10 LET trys = 0
70 FOR p = 1 TO 26
```



```

80 LET t (p) = p
90 NEXT p
100 FOR r = 1 TO 5
110 LET p = 1
120 READ m$
130 FOR c = 1 TO 31
140 IF m$ (p) = "□" THEN LET p = p + 1: GO TO 140
150 LET w$ (r, c) = m$ (p)
160 LET p = p + 1
170 NEXT c
180 NEXT r
190 FOR i = 1 TO 50
200 LET p = INT (RND * 26) + 1
210 LET q = INT (RND * 26) + 1
220 LET temp = t (p)
230 LET t (p) = t (q)
240 LET t (q) = temp
250 NEXT i
252 FOR i = 1 TO 26
253 LET t (i) = t (i) + 96
254 NEXT i
260 LET r = INT (RND * 5) + 1
270 FOR c = 1 TO 31
280 LET n$ (c) = CHR$ (t (CODE w$ (r, c) - CODE "a" + 1) )
290 NEXT c
300 PRINT AT 2, 2; "Here's the message . . ."
310 PRINT AT 10, 0; n$
320 INPUT "What's your decode?"; d$
330 LET c = 1
340 FOR p = 1 TO LEN d$
350 IF d$ (p) = "□" THEN GO TO 380
360 LET n$ (c) = d$ (p)
370 LET c = c + 1
380 NEXT p
385 LET trys = trys + 1
390 IF n$ < > w$ (r) AND trys < 2 THEN PRINT AT 2, 2;
    "Try again [15 spaces]": GO TO 320
400 IF n$ = w$ (r) THEN PRINT AT 2, 2; "You've got it!!!! □□□□□□□□"
410 INPUT "Want another one?"; q$
420 IF q$ = "yes" THEN CLS: RESTORE: GO TO 1

```

Other titles of interest

PEEK, POKE, BYTE & RAM! Basic Programming for the ZX81
Ian Stewart & Robin Jones

'Far and away the best book for ZX81 users new to computing'—*Popular Computing Weekly*

'... the best introduction to using this trail-blazing micro'—*Computers in Schools*

'One of fifty books already published on the Sinclair micros, it is the best introduction accessible to all computing novices'—*Laboratory Equipment Digest*

Machine Code and better Basic
Ian Stewart & Robin Jones

The ZX81 Add-On Book
Martin Wren-Hilton

Easy Programming for the ZX Spectrum
Ian Stewart & Robin Jones

Games to Play on your Spectrum
Martin Wren-Hilton

Available from October '82

Spectrum in Education
Eric Deeson

Easy Programming for the BBC Micro
Eric Deeson

Further Programming for the BBC Micro
Alan Thomas

Plus lots more! Keep your eye on the magazines for up-to-date news.

Can you:

- weave your way through the magic forest?
- ferry the wolf, the goat, and the cabbage across the river?
- share out the wine without spilling a drop?

All programs are designed to RUN on the Spectrum, and the first 15 are listed for the ZX81 with embellishments indicated to make use of the special Spectrum features.

Note: You'll need a 16K RAM pack for the ZX81.

Happy puzzling!

COMPUTERS HAVE BEEN A
PUZZLE TO ME FOR YEARS .

MY DEAR ...



cosgrove



Info in Spanish and English(down)

```
=====
S P A N I S H
=====
```

```
Título:      Computer Puzzles For spectrum & ZX81
Autor:       Ian Stewart y Robin Jones
Editorial:   Shiva Publishing LTD.
Año impresión: 1982
ISBN:        0-906812-27-5
Idioma, País: Inglés, Inglaterra
Nº de Páginas: 68 (60 Páginas numeradas + 4 sin numerar + portadas)
Contenido:   Introduce a la resolución de problemas a través de los puzzles.
Nivel:       Iniciado
Escaneado por: NEBIRE
```

Formato digital: *.CBZ (es un fichero *.ZIP, que puede visionarse directamente (sin descomprimir) con cualquier lector de cómic digital. Los hay para cada plataforma).

Pasos del proceso:

Acciones individuales (portadas aparte en *.bmp):

- Escaneo a 300ppp.
- Giro.
- Recorte.
- Guardar sin pérdidas (en *.RLE ya queda guardada a 256 colores y comprimida)

Acciones en lote (portadas aparte en *.jpg, páginas con gráficos llevan posterior acción):

Paso 1:

- Abrir
- Reducir a 256 colores (si no se hizo al guardar con el formato elegido).
- Ajustar tamaño para unificar todas (Excepto tapas).
- Variar brillo a -23 y contraste a +23
- Guardar como *.png: Árbol de octantes optimizado, basado en paleta de 32bits colores, sin transparencia. Luego...

Paso 2:

- Abrir
- Cambiar paleta actual por una de sólo 2 colores (elegidos 2 colores crema, claro para el papel y oscuro para la tinta. El excesivo brillo de color blanco, resulta molesto para la vista).
- Variar brillo a -16 y contraste a +9
- Guardar como *.png: Corte intermedio optimizado, basado en paleta de 1 bit colores, sin transparencia

Las imágenes con gráficos se guardan aparte.

Resultado: Imágenes de 1700*2700 px. de aprox. 50-80kb. cada imagen, a 2 tintas.

Paso 3: (sólo páginas con gráficos)

- Abrir: (la del paso 1 y la del paso 2)
- Cambiar paleta al par de imágenes, a una paleta de 16 tonos (2 de ellos son los mismos que la paleta anterior).
- Variar brillo a -16 y contraste a +9 (sólo a la imagen del paso 1).
- Cortar gráfico de imagen del paso 1 y pegarla en imagen del paso 2.
- Guardar como *.png: Corte intermedio optimizado, basado en paleta de 4 bit colores, sin transparencia

Resultado: La imagen queda comprimida en png, a 16 tintas, pero sólo se usan las 16 para el gráfico.

El tamaño final resultante varía según el tamaño del gráfico. si el tamaño es excesivo o fuera en color habría que pensar en otra solución.

- Escribir este archivo info.txt, comprimir en *.ZIP y renombrar a *.CBZ

```
=====
E N G L I S H
=====
```

Title: Computer Puzzles For spectrum & ZX81

Info.txt

Author: Ian Stewart y Robin Jones
Publisher: Shiva Publishing LTD.
Year release: 1982
ISBN: 0-906812-27-5
Language: English, UK
N° of Pages: 68 (60 numerated pages + 4 no numerate + covers)
Content: Introduce to problem solving through the puzzles.
Level: Initiated
Scanned by: NEBIRE
Digital format: *.CBZ (it's a *.ZIP file, You can watch directly (without decompress) with any comic book reader... There are for any platform).

Steps in the process:

Individual actions (separate covers in *.bmp):

- Scan to 300ppp.
- Rotate free.
- Clipping.
- Save as... losless (in *.RLE, then saved in 256 colors and compressed)

Batch actions (separate covers in *.jpg, the pages with graphics, pages with graphics take a subsequent action):

Step 1:

- Open
- Reduced to 256 colors (if it did not save with the chosen format).
- Adjust size to unify all (except covers).
- Adjust brightness to -23 and contrast to +23
- Save as *.png: Tree of octantes optimized, based on 32 bit palette colors, no transparency present. Then...

Step 2:

- Reopen
- Change current palette to a 2-colour (elected 2-colour cream, dark and light for the role for the ink. Excessive brightness of white, is annoying for the view).
- Adjust brightness to -16 and contrast to +9
- Save as *.png: Intermediate cut optimized, based on 1-bit palette colors, no transparency present.

Images with graphics are saved separately.

Result: Images of 1700*2700 px. approx. 50-80 kb. for each image, 2 colors.

Step 3 (only pages with graphics):

- Reopen (both; page in step 1, and step 2)
- Create a palette of 16 shades (2 of them are the same as the previous palette), and change current palette to the pair of images, for this.
- Adjust brightness to -16 and contrast to +9 (only to image on step 1).
- Cut graphic selection of image from step 1 and paste it into image of step 2 (this needs to be manually one by one).
- Save as *.png: Intermediate cut optimized, based on 4-bit palette colors, no transparency present.

Result: The image is compressed in png, 16 colors, but the 16 colors are used only for the graphic.

The resulting final size varies depending on the size of the graphic. If the size is excessive or in color would have to think in another solution.

- Write this file info.txt, compress in *.ZIP an rename in *.CBZ