USER NOTES 5

SHARPSOFT USER NOTES

Issue No. 5

This issue is mainly devoted to readers' contributions, letters and programs. Our request for hints, programs and comments resulted in plenty of feedback from subscribers. To date we have reached a figure of roughly 700 subscribers, which once again allows the user notes to be self-supporting. To all our readers who have either written to SHARPSOFT or phoned, our thanks for your contributions to us and we will do our best to publish them. However, please don't be disappointed if your contribution is not published for a few months. Each issue seems to get longer! In fact, we have had to limit the size of each issue to roughly 100 pages so that we do not exceed our budget.

Our articles in this issue include information on FORTH, an extended editor, Tiny c, languages in general and notes on assembler input-output routines. In some cases these articles refer to other articles which, due to space, will appear in future user notes. The major new series on CP/M and hardware is likely to start in the next issue, space permitting.

The SHARP PASCAL package appears to be generating a lot of interest among users. Included in both the letters and programs section you will find readers' PASCAL programs and hints on using the PASCAL inpterpreter.

Once again, your letters and comments contain most valuable information for the SHARP computer user. Personally, I believe the exchange of ideas among users is one of the most valuable functions these user notes provide - so please keep writing to us with your ideas.

Mike Brinson

Editor

FORTH - Bugs and the EDITOR
by Mike Brinson

Since the last issue of these user notes we expect many readers have tried programming in FORTH. Some of you will have realised the power of the language as soon as you loaded the FORTH tapes. Others are probably still puzzled by the FORTH syntax! The following notes are included in this issue to help everyone obtain the best from their FORTH language system.

A number of readers have either phoned or written to us asking for more information on how to load the EDITOR and ASSEMBLER tapes. If you are still having trouble loading your tapes then the following notes will help you.

Due to the hardware limitations of the cassette based MZ-80K computer when writing the virtual memory operating system for FORTH we were forced to adopt the following scheme:-

1. Each FORTH screen is recorded as a program on tape with a header name - the screen number.

2. The sequence of screen numbers must be sequential - 1 at the beginning of the tape.

3. Tapes must be FORMATTED before FORTH screens can be recorded - to ensure screens are NOT recorded in the wrong sequence - overwriting other screens.

4. Each FORTH screen in recorded as a 1024 byte block - using the SHARtape format.

These requirements imply that if you wish to LIST or LOAD a screen with a high header number, the operating system will have to read over ALL the lower numbered screens - THIS DOES TAKE TIME! The positions of the screens on a tape are roughly as follows:-

Tape Counter	Screen Number
0	1
25	2
35	3
45	4
55	5
65	6
75	7
85	8
etc.	etc.

Hence to LIST EDITOR Screen 7 - first load the FORTH languages tape using the monitor. Replace the FORTH tape with the EDITOR tape and rewind - set the tape counter to 0 and fast forward the tape to roughly counter position 70. Then type:-

DECIMAL 7 LIST (CR)

Screen 7 should then be listed on the VDU. LOADing the EDITOR uses a similar sequence - NOTE - because the operating system loads the complete editor program (roughly 5 screens) there is a delay before the O.K. prompt is displayed.

Notes on using the EDITOR

If you wish to write and save programs on tape then the FORTH editor must be used for this purpose. Remember 'FORMAT' the tape you are using for program storage BEFORE you attempt to save any screens.

Once the EDITOR is loaded - typing EDITOR (CR) makes the EDITOR vocabulary the first vocabulary searched when FORTH is operating on words entered from the keyboard.

The EDITOR provided with your FORTH tapes is a simple to use line oriented editor. It allows lines to be entered, changed and deleted and the first screen saved on tape for later compiling and use.

Each screen consists of 16 lines (numbered 0 to 15) with 64 characters per line. Additionally, there is a single line of temporary storage (displayed at the bottom of the VDU) where information can be held when moving lines from one position to another - this line is called the PAD.

EDITOR commands

P n---
Put the following text on line n. Write over its old contents.

I n---
Insert text from the PAD to line n. Shift the original nth and subsequent lines down by one line. The last line in the screen is lost.

R n---
Replace the nth line with the text stored in the PAD.

E n---
Erase the nth line in the scren by filling with 64 blanks.

D n---
Delete the nth line. Move subsequent lines up one line. The deleted line is held in the PAD incase it is still needed.

H n---
Copy the nth line to the PAD. Hold the text there ready for later use.

S n---
Spread nth line with blanks. Down shift the original nth and subsequent lines by one line. The last line in the screen in lost.

T n---
Type the nth line in the current screen. Save the text in the PAD.

L Re-list the screen being edited.

CLEAR n---
Clear the nth screen by padding with blanks.

COPY n1 n2 ---
Copy screen n1 to screen n2.

The above words all apply to line editing. However, if some errors are discovered or only a few characters in a line need to be changed, the line editor is not suitable because one would need to re-type the complete line. For single character changes - a string editor is more effective. The following FORTH screens present an editor of this form - we call this FIG. FORTH EDITOR V2.0.

```
7 LIST
SCR # 7
  0 ( FIG-FORTH EDITOR V2.0 SCREEN 1 )
  1 FORTH DEFINITIONS HEX
  2 : TEXT HERE C/L 1+ BLANKS WORD
  3        HERE PAD C/L 1+ CMOVE ;
  4 : LINE DUP FFF0 AND 17 ?ERROR
  5        SCR @ (LINE) DROP ;
  6 : 2DROP DROP DROP ;
  7 : 2SWAP ROT >R ROT R> ;
  8 VOCABULARY EDITOR IMMEDIATE HEX
  9 : WHERE DUP B/SCR / DUP SCR !
 10         ." SCR # " DECIMAL .
 11         SWAP C/L /MOD C/L * ROT
 12         BLOCK + CR C/L TYPE CR
 13         HERE C@ - SPACES 5E
 14       EMIT [COMPILE] EDITOR
 15       QUIT ;    -->
```

```
8 LIST
SCR # 8
  0 ( FIG-FORTH EDITOR V2.0 SCREEN 2 )
  1 EDITOR DEFINITIONS HEX
  2 : #LOCATE R# @ C/L /MOD 1
  3 : #LEAD #LOCATE LINE SWAP 1
  4 : #LAG #LEAD DUP >R + C/L R> - ;
  5 : -MOVE LINE C/L CMOVE UPDATE ;
  6 : H LINE PAD 1+ C/L DUP PAD C! CMOVE ;
  7 : E LINE C/L BLANKS UPDATE ;
  8 : S DUP 1 - 0E DO I LINE I 1+
  9    -MOVE -1 +LOOP E ;
 10 : D DUP H 0F DUP ROT DO I
 11      1+ LINE I -MOVE LOOP E ;
 12 : M R# +! CR SPACE #LEAD TYPE 5F
 13      EMIT #LAG TYPE #LOCATE . DROP ;
 14 : T DUP C/L * R# ! DUP H 0 M ;
 15 -->
```

```
9 LIST
SCR # 9
  0 ( FIG-FORTH EDITOR V2.0 SCREEN 3 )
  1 : L SCR @ LIST 0 M ;
  2 : R PAD 1+ SWAP -MOVE ;
  3 : P 1 TEXT R ;
  4 : I DUP S R ;
  5 : TOP 0 R# ! ;
  6 : CLEAR SCR ! 10 0 DO FORTH I
  7         EDITOR E LOOP ;
  8 : COPY B/SCR * OFFSET @ + SWAP
  9       B/SCR * B/SCR OVER + SWAP
 10       DO DUP FORTH I BLOCK 2 - !
 11       1+ UPDATE LOOP DROP FLUSH ;
 12 -->
 13
 14
 15
OK


10 LIST
SCR # 10
  0 ( FIG-FORTH EDITOR V2.0 SCREEN 4 )
  1 : -TEXT SWAP -DUP IF OVER + SWAP
  2         DO DUP C@ FORTH I C@ -
  3         IF 0= LEAVE ELSE 1+ THEN LOOP
  4         ELSE DROP 0= THEN ;
  5 : MATCH >R >R 2DUP R> R> 2SWAP
  6         OVER + SWAP DO 2DUP FORTH
  7        I -TEXT IF >R 2DROP R> - I
  8         SWAP - 0 SWAP 0 0 LEAVE THEN
  9         LOOP 2DROP SWAP 0= SWAP ;
 10 : 1LINE #LAG PAD COUNT MATCH R# +! ;
 11 : FIND BEGIN 3FF R# @ < IF TOP
 12        PAD HERE C/L 1+ CMOVE 0
 13        ERROR ENDIF 1LINE UNTIL ;
 14 -->
 15
OK


 11 ;LIST
 SCR # 11
   0 ( FIG-FORTH EDITOR V2.0 SCREEN 5 )
   1 : DELETE >R #LAG + FORTH R -
   2           #LAG R MINUS R# +!
   3           #LEAD + SWAP CMOVE
   4           R> BLANKS UPDATE ;
   5 : N  FIND 0 M ;
   6 : F  1 TEXT N ;
   7 : B  PAD C@ MINUS M ;
   8 : X  1 TEXT FIND PAD C@ DELETE
   9      0 M ;
  10 : TILL #LEAD + 1 TEXT 1LINE
  11        0= 0 ?ERROR #LEAD
  12        + SWAP - DELETE 0 M ;
  13 -->
  14
  15
 OK
```

```
12 LIST
SCR # 12
  0 ( FIG-FORTH EDITOR V2.0 SCREEN 6 )
  1 : C 1 TEXT PAD COUNT
  2     #LAG ROT OVER MIN >R
  3     FORTH R R# +! R - >R
  4     DUP HERE R CMOVE HERE
  5     #LEAD + R> CMOVE R>
  6     CMOVE UPDATE 0 M ;
  7 FORTH DEFINITIONS DECIMAL
  8 LATEST 12 +ORIGIN !
  9 HERE 20 +ORIGIN !
 10 HERE 30 +ORIGIN !
 11 ' EDITOR 6 + 32 +ORIGIN !
 12 HERE FENCE ! !S
 13
 14
 15
OK
```

This editor contains all the line editor commands plus the following character editing commands.

TOP — Move the cursor to home, top left of the screen.

FIND — Search the entire screen for a string stored in the PAD. If not found, issue an error message. If found, move the cursor to the end of the found string.

DELETE — n---
Delete n characters in front of the cursor. Move the text lfrom the end of the line to fill up the space. Blank fill at the end of the line.

N — Find the next occurrence of the text already in the PAD.

F — Find the first occurrence of the following text string.

B — Back the cursor to the beginning of the string in the PAD.

X — Delete the following text from the current line.

TILL — Delete all characters from cursor location to the end of the following text string.

C — Spread the text at the cursor to insert the following string. Characters pushed off the end of the line are lost.

Bugs

One minor bug has so far come to light in our FORTH package. Most of you will probably have noticed that error messages and the output from VLIST include graphics symbols! This makes the VLIST output particularly hard to read. FIG. FORTH is defined using the ASCII characters set (codes 0 to 127). However, on the MZ-80K SHARP included graphics symbols using codes 128 to 255. Hence, characters with the high order bit set to one are printed as graphics. Also in FIG. FORTH the last letter in a word header name has the highest bit set - hence the graphics character output in VLIST. The word which causes this problem is ID. One solution is to write a new VLIST with ID. changed.

The FIG. FOTH Glossary includes the word DUMP - this is NOT included on your FORTH tapes. The specification for DUMP and its code are:-

DUMP addr n---
Print the contents of n memory locations beginning at addr. Both addresses and contents are shown in the current numeric base.

```
: DUMP
        0 DO CR 8  R

        8 0 DO DUP @ 8  R

        2+ LOOP 8 +LOOP DROP;
```

If you think you have found a bug in FORTH please write to us and we will check it out. More FORTH in the next issue.

Languages
-
Interpreters, Compilers and Assemblers

Every month SHARPSOFT receives enquiries from MZ80K and MZ80B users asking about language software for these computers. These questions range from " Is this or that language available?" to more complex technical queries regarding the suitability of a particular language for a given application. This short article is an attempt on our part to provide User Note readers with information loosely connected with the subject of computer languages and their availability on the MZ80K and MZ80B computers. If you would like to add your comments to the discussion of languages write to us and we will publish replies next issue.

For most users BASIC is still the main applications programming language. 1981 was the tenth anniversary of the introduction of the microprocessor. BASIC dominated this first decade. The popularity of BASIC originates from two main factors; firstly it is an easy language to learn and is relatively user friendly, secondly it requires the minimum of computer hardware and software to implement it on a small computer. Nearly all of the BASIC's supplied with microcomputers are interpreters rather than compilers.

A BASIC interpreter, as the name implies, translates the BASIC keywords into actions which the computer undertakes directly. The BASIC keywords are never converted into machine code. Usually a BASIC interpreter includes within its structure some form of editor which is used for program preparation and correction of logical and typing errors.

The interpreter approach to high-level computer languages has a number of distinct strengths and weaknesses. On the positive side - it provides good interaction between the user and the computer making program development straightforward and fast. This is very important for the new programmer who is often learning a computer language for the first time. On the negative side - the interpreter must reside in memory with the users program, reducing the available memory space by a significant amount. Also execution speed deteriorates as the user's program size increases, particularly when a program includes long loops and a large number of variables or arrays.

In the high-level language league table BASIC is the poor relation. Its structure and language facilities offer users only a minimum of those features found in other languages. If you are writing small programs, lets say 100 lines of BASIC, this probably does not matter. However, as programs grow in size problems develop. With larger programs clarity, readability and structure of the program become important. Unfortunately, the very interactive nature of BASIC allows inexperienced users to "add-on" sections of program, as a program is being developed,

without much thought to the overall structure of the program. In such cases each component part is often connected by GOTO statements. This approach is fine if your program works, but, if the program does not work finding the error becomes a programmer's nightmare.

Please don't misunderstand the above comments - BASIC is an important computer language and will for many years to come remain one of the main languages first learned by newcomers to computing.

Usually after becoming proficient in BASIC most programmers begin to realize the languages strengths and weaknesses. Programs often run too slowly and need to be optimized for speed which in turn destroys the clarity of the program code.

Speed critical sections of a BASIC program can be replaced by machine code segments through the popular DATA,PEEK,POKE and USR constructions. This is a worthwhile approach when speeding-up programs but does require a knowledge of Z80 machine code. Ultimately, maximum program run speed is obtained by writing programs entirely in Z80 assembly code. However, the effort needed to write a program in assembler language is normally orders of magnitude greater than BASIC.

During the last decade professional programmers have tended to replace assembly code by high-level languages where the language is NOT interpreted but compiled to machine code. The program which undertakes this translation process is generally called a compiler. Compilers are begining to emerge into the microcomputer scene. The benefits are two-fold; firstly it is easier to write and maintain a large program in a high-level language than in assembly code,secondly with a quality compiler program run speeds approaching assembly code speed are possible. An important extra bonus is that compiled languages, for example PASCAL and C, provide the user with good structured constructions which allow excellent readability, program flow control and error checking.

One disadvantage to the compiler approach to high-level languages is however, that compilers are normally more difficult to use than an interpreter and do often require more hardware than is provided by a basic microcomputer.

Getting back to the original question "What language software is available?", leads one to consider the type of application the language is to be used for. Starting with the MZ80K as a base level allows users to expand along a number of different paths, for example:-

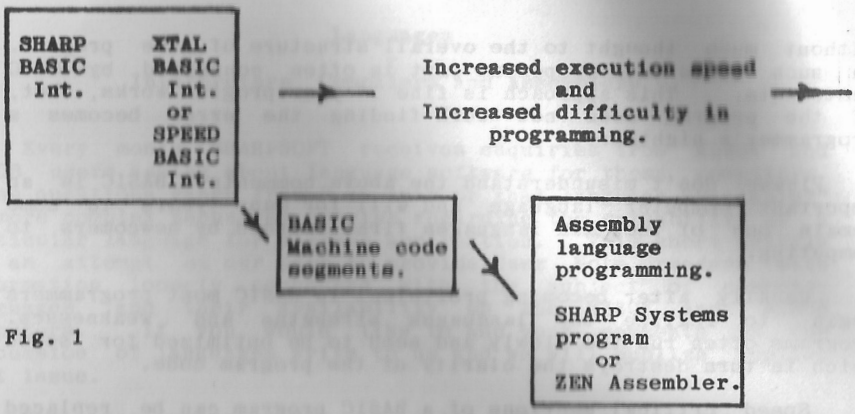


Fig. 1

The language route shown in Fig. 1 is the classical single user hobbyist path. It has the advantage of providing a good range of software at reasonable cost and does not require a floppy disc based computer system. Tape based structured languages can also be added to your language software, see Fig. 2.

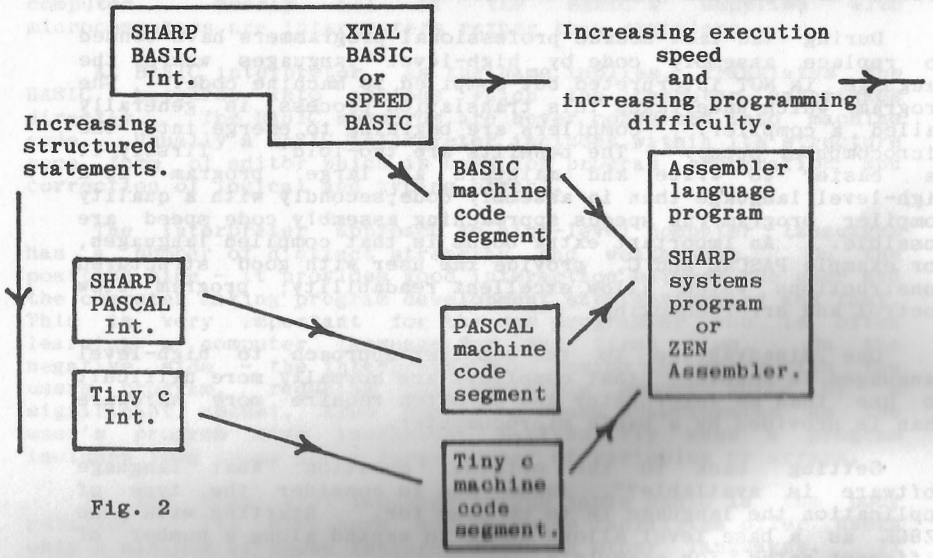


Fig. 2

Fig. 2 requires some explanation. SHARP PASCAL is similar in execution speed to SHARP BASIC but is a structured language with better readability and clarity of code. Although NOT a full implementation of standard PASCAL it allows users with tape based MZ80K computers to learn and experiment with a structured language. Full marks to SHARP for developing this package. In

our opinion this package is a step in the right direction towards better software for the MZ80K. Again, if required, run speed can be improved by using machine code program segments OR by using integer variables instead of real variables.

If code elegance is your objective then Tiny c is for you. Similar to SHARP PASCAL the Tiny c language is strong on structuring and clarity but does tend to run slower than the other interpreters. Machine code routines will help to improve execution performance. Interfacing machine code routines to Tiny c is much easier than BASIC. Tiny c allows routines to be given descriptive names rather than a USR call.

Readers will have noticed that FORTH is missing from Figs. 1 and 2. This was done deliberately because FORTH tends to contain the major features of all aspects of these diagrams. FORTH being an extensible language allows extensions in any direction the user wishes. FORTH is one of the most exciting and significant developments in the field of microcomputer languages to have appeared in the last few years. One criticism, against FORTH is its poor readability. The "reverse polish syntax" comes hard after the elegance of PASCAL and Tiny c !

Choosing a language for your particular needs is a difficult task - perhaps the following points may help. BASIC and PASCAL are ideal for applications which contain a significant amount of of calculations, PASCAL has the more superior approach to structuring and readability. However, for applications where speed is important, such as games with graphics, machine code is much faster. Tiny c's strength lies in its code elegance and is very good for teaching structured programming and is often used for developing systems programs.

If your finances allow then investment in both the SHARP PASCAL and Tiny c interpreters will enhance the enjoyment and use of your computer. The applications of FORTH are really up to you !!!! -- limited only by your ideas.

Excluding FORTH all the implementations of the languages mentioned above are interpreters. To complete the computer languages options for the basic MZ80K computer, compiler versions of these popular languages are needed. Ideally these compilers should produce Z80 machine code. This is not a simple task given the limited hardwhere resources of the MZ80K. At SHARPSOFT we are attempting to tackle this problem by developing a tape based PASCAL COMPILER for the MZ80K. Our objective is to develop a compiler for a sub-set of the PASCAL language. This compiler could then be used to replace assembly code programming where speed is important. If this development is successful we will be including the PASCAL compiler free with next years user notes subscription. More details on the progress of this project will be included in the next issue of these user notes.

Compilers are easier to implement on computers which are equiped with floppy discs and a respectable operating system. For the MZ80k or MZ80B owner two routes are possible. Firstly floppy discs must be added to your computer. In the case of the MZ80K the cheapest upgrade is to first add a single disc drive, then if finances allow to add a second drive at a later time. Unfortunately, for the home user this represents a significant investment, so it becomes very important to select the right language software for your computer. Assuming that you have upgraded your computer to include a disc or discs then a disc operating system must be selected for use with the disc drive unit. At the present time there are two popular disc operating systems available for both the MZ80k and MZ80B computers. Firstly, there is SHARP'S FDOS system and secondly the more universally known CP/M operating system. Both have advantages and disadvantages for the small computer user.

SHARP introduced FDOS towards the end of last year. This operating system is a complete package for use with either single or dual disc drive units and includes features for all the standard disk "house-keeping" such as copying and deleting disc files. Included in the package is an editor, for program preparation, and a Z80 assembler with linking loader and debugger which is similar to the tape based Systems programs. Overall this is an impressive package ideally suited to the needs of the assembly code programmer.

The first software released by SHARP for use with FDOS is a BASIC compiler. This compiler is an excellent piece of language software generating Z80 code from BASIC which runs at 2 to 5 times faster than the SHARP BASIC interpreter. All the major language features, including floating point arithmetic, are retained. As an added bonus the BASIC syntax has been extended to allow long variable names and named machine code segments. When coupled with the facilities in FDOS, the low cost of this compiler makes it an outstanding software development and a very good investment for those upgrading there computer system.

Unfortunately, at the present time no other language software appears to be available to run under FDOS. We expect this situation to change as more users get to grips with the FDOS package. If demand is sufficient we at SHARPSOFT may develop a version of FORTH to run under FDOS. Incidently this would involve us in three to six months work and at the moment we just don't have time to undertake such a task.

For SHARP FDOS represents a major software development and in the future one would expect to see quality language software developed for use with their disc operating system; perhaps a PASCAL or FORTRAN compiler - one can only speculate !!!!

For small computer users the CP/M operating system probably represents the ultimate in sophistication. Its great strength lies in the availability of a wide range of language and other software. Whether you are a home user or a small business user

there is something for everyone. For example, the CP/M user library contains roughly 70 eight inch floppy discs full of software. After conversion to SHARP disc format this software is available at roughly media cost. A number of compilers are included in the library, for example compilers for the ALGOL,BASIC and PASCAL languages. The quality of this software is variable but considering each disc only costs a few pounds each volume represents outstanding value. The universal acceptance of CP/M for 8080,8085 and Z80 microcomputers has encouraged software houses to develop impressive language packages for us under CP/M. Typical examples are Digital Research with their PL/1 compiler, Microsoft with their BASIC and FORTRAN compilers and ITHACA Intersystems with their PASCAL/Z compiler. Software of this quality is expensive ranging from approximately 100 pounds to 600 pounds for the most expensive CP/M items. Cost is the single greatest limit to the wider acceptance of CP/M among home users. However, for the small business user or scientific user the development time saved by use of these high quality packages often repays the initial investment. A more detailed background and discussion of CP/M on the MZ80K and MZ80B computers is given in our new CP/M column.

To summarize a wider range of language software is available for the MZ80 computers than is generally realized by the average SHARP user. Today, there is something for everyone whether you own a basic MZ80k computer or a hard-disc based MZ80B system. As the number of SHARP users grows the amount of low cost good quality software will also increase. Compared to even a year ago the software has improved at faster rate than we would have predicted. Lets hope this trend continues !!!!!

TINY C TUTORIAL

These tutorial notes attempt to introduce readers to the Tiny c language. Tiny c is a subset of the C language developed at the Bell Telephone Laboratories by Dennis M. Ritchie. We hope that this article will give you an insight to this exciting package.

We would like to thank Irene and Tom Gibson of Tiny C Associates for their help in the preparation of this user notes contribution. ED.

Tiny c notes.

1. Tiny c is a structured programming language.

2. Programs are built using modules. Each module usually does one simple task.

3. When programming in tiny c it is:

   (a) easier to debug programs, and

   (b) easier to design large programs.

4. Tiny c is a must for programmers who write large programs in BASIC and who have experienced debug problems.

5. A simple example program

```
Example
[
 pl "My first tiny c program"
]
```

6. The general structure of a tiny c program

```
Name
[
program body
]
```

7. To execute (run) program "Example"

```
>. Example <cr>
My first tiny c program
>
```

8. A second example program

```
Printlines
[
 pl "Tiny c runs well on the MZ80K"
 pl "Implementation by SHARPSOFT"
 pl "Tiny c is licensed by Tiny C Associates"
]
```

9. Running Printlines

```
>. Printlines
Tiny C runs well on the MZ8ØK
Implementation by SHARPSOFT
Tiny C is licensed by Tiny C Associates
>
```

10. The Tiny c program preparation system (pps)

The pps includes

(a) a text editor
(b) a run time environment, and
(c) a library of standard functions.

The pps prompt is >

No LINE NUMBERS !!!!!

--- a pure text editor ---

11. pps

```
> First line of text
> Second line of text
> Third line of text
>. l
 First line of text
>. p 3
 First line of text
 Second line of text
 Third line of text
>
```

12. Current line

At any time one line is the CURRENT line

```
>. p      prints it
>. 7      makes line 7 current
>. +3     makes line 1Ø current
>. -6     makes line 4 current
```

13. If you get lost when using the editor

```
>. /      prints four numbers:

          2    3    47   3953
          |    |    |     |
          |    |    |     free memory
          |    |    used memory
          |    total number of lines
          current line number
```

14. More pps practice

```
>. 0            (set line editor pointer to line 0)
>. p 99         (prints 99 lines)
First line of text
Second line of text
Third line of text
>. /
3    3    70   3930
```

15. Inserting text

New text is always inserted AFTER the current line.
New text BECOMES the current text.

```
>. 1
 First line of text
> Inserted second line of text
> inserted third line of text
>. 0
>. p 99
 First line of text
 Inserted second line of text
 Inserted third line of text
 Second line of text
 Third line of text
>. /
5    5    143  3860
```

16. Deleting text

```
>. d      deletes the current line

>. d 3    deletes the current line and the next two lines

>. p
 Third line of text
>. d
>. p
 Second line of text
> Fifth line of text
>. 0
>. p 99

 First line of text
 Inserted second line of text
 Inserted third line of text
 Second line of text
 Fifth line of text
```

17. Review of the pps so far

>.p [n]	prints n lines. Makes last line the current line.
>.d [n]	deletes n lines. Makes previous line the current line.
> text	inserts text after current line. Text is any line NOT starting with .,+ or -.
>./	prints the currentline, total number of lines, used memory space, and free memory space.
>. number	makes line "number" the current line.
>+	moves down one line.
>+ number	moves down "number" of lines.
>-	moves up one line.
>- number	moves up "number" of lines.

18. Changing text

```
>.p
 Third line of text
>.c/Third/Fourth/
 Fourth line of text
```

NOTES.
Only the current line is changed.
If the old text is found it is replaced by the new text.
Otherwise no change is made.
The line is always printed.

19. Locating text

```
>.Ø
>.l/Second/
 Second line of text
```

NOTES.
Starts with line AFTER current line.
Searches down buffer until the line containing text to be located is found.
Makes that line current.
Prints it.
If text is not found,the current line is not changed,and a '?' is printed.

20. Saving programs on tape (or disc)

```
>.w filename
      |
      |
      must be a new name for each program.
```

21. Deleting a program from memory

```
>.1              (set text pointer to line 1)
>.d 9999         (delete 9999 lines-any big number is ok)
>./
0    0    0     4000
                  |
                  |
                 free space
```

22. Reading programs from tape (or disc)

```
>.r filename
      |
      |
     name of program to be read from tape or disc
```

The Tiny C language

1. Constants

```
7
     integer constants
-3

'a'
'X'  character constants
' '

"hello"
"This is a sentence"     String constants
""
integer max 16767
        min -16768
character 'one character'
string    "zero or more characters"
```

NOTE "x" and 'x' are different.

2. Names

Names are used for variables and functions.
One or more characters long.
First is alpha, the rest may be alphanumeric.

```
k7          hello
MARY        mike
testl       begood

cat         CAT         different names

The first 7 and the last character are significant.
catinthehat     catintheclosit       same names
```

3. Variables

```
int value(5)

value(Ø)    to    value(5)
```

4. Expressions

```
7+3              1Ø
(7+3)/3          3
'a'+2            'c'
9<11             true (1)
11<9             false (Ø)

char letter
letter = 'x'
letter<'z'       true (1)
```

5. Operators

```
+ - * / %
< <= > >= == !=
=    for assignment only

%    means remainder after division
     1Ø % 3 is 1
     7  % 4 is 3

!=   means NOT EQUAL
     7 != 2 is true (1)
     7 != 7 is false (Ø)

==   means EQUAL TO
     7 == 7 is true (1)
```

6. Assignment

```
a=b=c=Ø
x(k=k+1) = a-(b=c/d)

NOTE
Assignment and equality are different.
x=Ø       put Ø into x
x==Ø      test if x is equal to Ø.
```

7. If statement

```
if (condition) statement
```

examples

```
if(k<10)x=0
if((a+b)/c==10) ps "ten"
if(k<10) x(k=k+1) = a-(b=c/d)
if(letter>='a')
  if(letter<='z')
     letter=letter-32
```

8. Compound statements

```
if(k<m+7) [
           k=k+1
           m=m-1
           diff=m-k
          ]
```

Any statement can be a compound statement

9. While statement

```
while ( condition ) statement
```

Examples

```
while(k<10) x(k=k+1)=a-(b=c/d)
while(letter=buff(i=i+1))
      if(letter>='a')
        if(letter<='z')
          buff(i)=buff(i)-32
while(k<m+7)   [
                k=k+1
                m=m-1
                diff=m-k
               ]
```

10. Strings

```
char buff(64)
ps   "type your name"
gs   buff
```

NOTES
gs is a function which reads a string into buff.

```
buff -- Tom'null'************
```

Strings are addresses of buffers.
The buffers have zero or more characters followed by a null.

11. Comparing strings

```
WRONG way

buff=="hello"

RIGHT way

ceqn (buff,"hello",5)
 |
 |
 Character string equals function.
```

NOTES
When testing strings use function ceqn.
When testing integers or characters use ==.

12. Printing characters,strings and numbers

```
pl "string"
pl buff
ps "string"
pn number
```

Printing uses library functions.

13. Reading characters, strings and numbers.

```
char buff(64)
int k,length

k=gn()                    gets a number from the keyboard.
length=gs(buff)           gets a string from the keyboard,
                          puts string in buff,
                          and returns length.
```

Reading uses library functions.

14. Simple program

```
guess
[
 char answer(64)
 ps "What is my name ? "
 gs(answer)
 if( ceqn(answer,"tiny c",6)) ps "RIGHT"
     else ps "WRONG"
]
```

15. Do forever

```
while (1) [
            .
            .
            if ( condition ) break
            .
            .
          ]
```

NOTES
break quits INNERMOST while

16. Innermost while

```
while (1) [
            .
            .
            while(1) [
                .
                .
              if( something ) break
                                  |
                                  |
                        ]         |
            .    ------ goes here
          ]
               ------ not here
```

17. if-else

```
if ( condition ) statement
else statement
```

Examples

```
if(k<limit) total=total+x(k)
else break
if(a<b+c) [
           pl "enter a bigger a "
           a=gn
         ]
else pl"you got it"
```

NOTES
On one line use ; before else

```
if( condition ) statement ; else statement
```

Example

```
if(gameover) score ; else yourturn
```

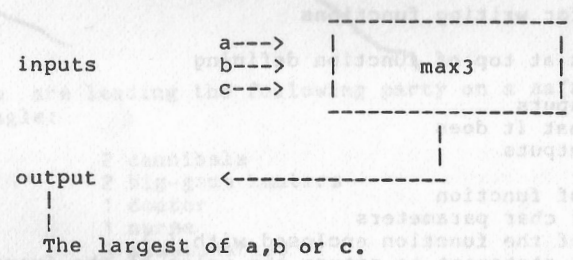
18. Alternation

```
if( a ) st1
else if( b ) st2
else if( c ) st3
else if( d ) st4
else stlast
```

NOTES
One and only one statement is done.
First to evaluate true gets done.
If none true, stlast is done.
Any of the 'st' can be
(a) simple
(b) compound --- many lines long
(c) complex --- if-else-while
For example:-

```
editor
[
 char     buff(32),c
 while(1) [
           ps">"
           gs(buff)
           c=buff(0)      /* command letter
           if(c=='p') ps"print"
           else if(c=='c') ps"change"
           else if(c=='d') ps"delete"
           else if(c=='l') ps"locate"
           else if(c=='x') break
           else ps "?"
          ]
]
```

19. Functions --- the black boxes of software



The largest of a,b or c.

```
max3 int a,b,c
[
 int m
 if(a>b) m=a ; else m=b
 if(c>m) m=c ; m=c
 return m
]
```

USING max3

```
main
[
 int x,y,z
 ps"enter three numbers " ; pl""
 x=gn ; y=gn ; z=gn
 ps"The largest of these numbers is "
 pn max3(x,y,z)
]
```

CALLING a function

A function may be used anywhere you can use a variable,for example

```
7+max3(a,b,19)/3
```

RULES

(a) write the functions name
(b) provide the correct number of arguments
(c) use() around the arguments.

TYPE of ARGUMENTS

Use strings or buffers with strings for string arguments,for example

```
pl"hello" ; pl"" ; pl buff
```

Integers and characters are interchanged for integer and character arguments,for example

```
pn 'A'    /* prints 65
```

Constants and variables are interchanged,for example

```
pn 7 ; pn x
```

20. Techniques for writing functions

Put comments at top of function defining

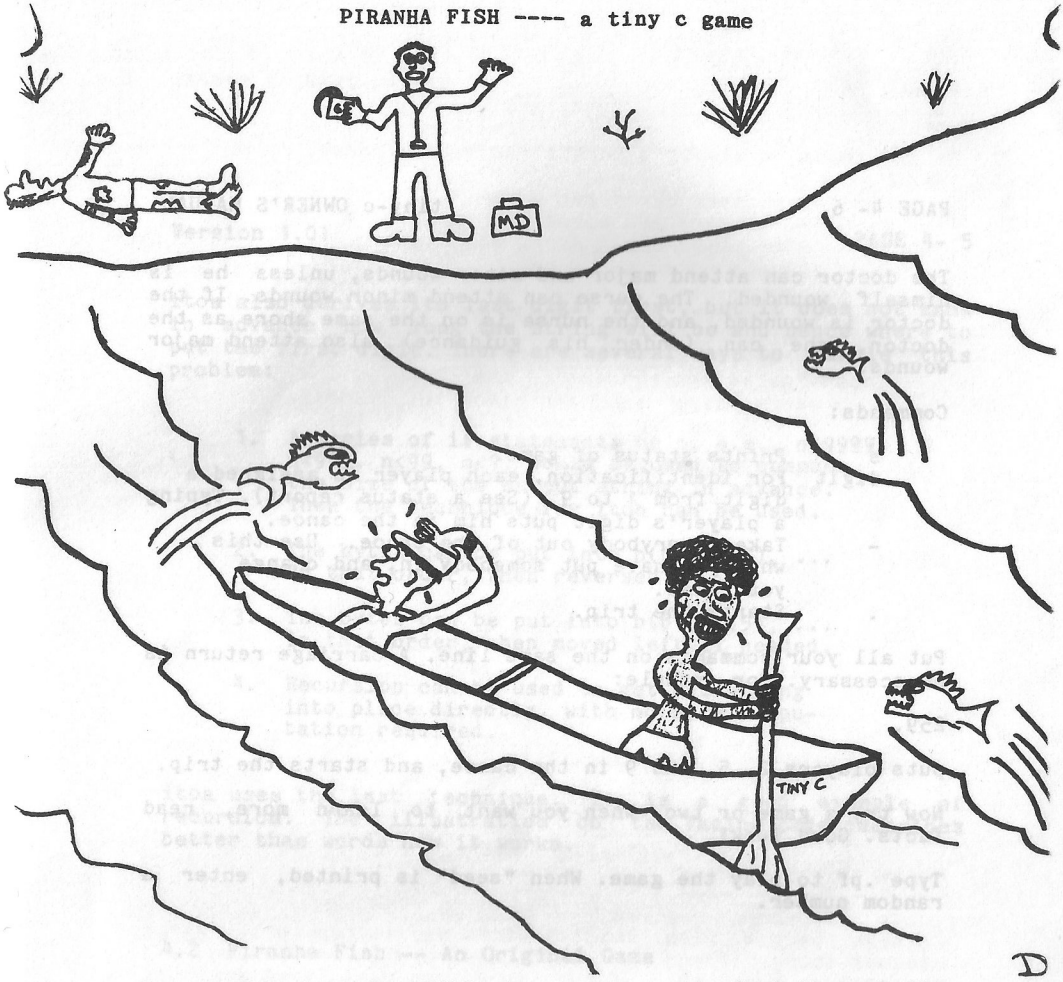
(a) inputs
(b) what it does
(b) outputs

Write name of function
Write int or char parameters
Write body of the function enclosed with [ ]
Use a return statement to return the value of the function.

21. Standard functions

Standard library	25 functions
Optional library	7 functions
Private library	as many functions as you care to write

PIRANHA FISH ---- a tiny c game



You are leading the following party on a safari through the jungle:

2 cannibals
2 big-game hunters
1 doctor
1 nurse
3 missionaries

You arrive at a 100-yard-wide river filled with piranha fish. You must cross the river. There is a leaky canoe on your shore, which can hold, at most, 4 people. The cannibals paddle the best, followed by the hunters, the doctor, the nurse, and the missionaries, who are notoriously weak. You must decide who gets in the canoe for each trip back and forth. Get the party across with a minimum of carnage.

The doctor can attend major and minor wounds, unless he is himself wounded. The nurse can attend minor wounds. If the doctor is wounded, and the nurse is on the same shore as the doctor, she can (under his guidance) also attend major wounds.

Commands:

s	Prints status of game.
digit	For identification, each player is assigned a digit from 1 to 9. (See a status report). Typing a player's digit puts him in the canoe.
-	Takes everybody out of the canoe. Use this when you have put somebody in, and change your mind.
.	Starts the trip.

Put all your commands on the same line. A carriage return is unnecessary. For example:

259.

puts players 2, 5, and 9 in the canoe, and starts the trip.

Now try a game or two. When you want to learn more, read facts. Good luck!

Type .pf to play the game. When "seed" is printed, enter a random number.

4.2.1 Facts

The speed of the canoe is the average of the paddling strengths of the players in the canoe. A speed of 100 gets the canoe to the opposite shore just as it fills.

The initial paddling strengths are:

cannibals	120
hunters	90
doctor	70
nurse	50
missionaries	40

itoa also derives its last digit first, but it does not know in advance how long the string will be and hence where to put the first digit. There are several ways to handle this problem:

1. A series of if statements on n, e.g., n<9999, n<999, n<99, n<9, could be used to compute tne size of the output string in advance. Then the technique for itoh can be used.

2. The bytes can be put into b(0), b(1), ... in that order, then reversed.

3. The bytes can be put into b(6), b(5), ... in that order, then moved left if needed.

4. Recursion can be used to get everything into place directly, with no size computation required.

itoa uses the last technique. It is a good example of recursion. The illustration on the facing page describes better than words how it works.

4.2 Piranha Fish -- An Original Game

You are leading the following party on a safari through the jungle:

2 cannibals
2 big-game hunters
1 doctor
1 nurse
3 missionaries

You arrive at a 100-yard-wide river filled with piranha fish. You must cross the river. There is a leaky canoe on your shore, which can hold, at most, 4 people. The cannibals paddle the best, followed by the hunters, the doctor, the nurse, and the missionaries, who are notoriously weak. You must decide who gets in the canoe for each trip back and forth. Get the party across with a minimum of carnage.

On the shore, a player's health can become worse:

Healthy players never get worse.
Attended players get worse with prob 0.11.
Unattended players get worse with prob 0.33.
Dead players never get worse.
To get worse means a minor wound becomes major, or a player with a major wound dies.
When a minor attended wound gets worse, it becomes a major unattended wound.
These "worse health" events are computed for every player once per canoe trip, whether or not the player participated in the latest trip. So players wounded early have more chances to get worse than players wounded later.

Score:

1000 for a perfect game.
-100 per dead player.
-30 for major unattended wounds.
-15 for major attended wounds.
-10 for minor unattended wounds.
-5 for minor attended wounds.

Highest score achieved to date is 995.

Maximum Carnage Game

Certain people with twisted minds may decide to try for a minimum score. If you do this, a new rule is needed: On each successive round trip of the canoe, you must leave at least one more person on the far shore than on the previous round trip.

Happy paddling!

Strengths are multiplied by the following factors for unhealthy paddlers:

minor wound, attended	0.9
major wound, attended	0.8
minor wound, unattended	0.8
major wound, unattended	0.7
dead	0.0

During a trip certain events happen, with probabilities shown:

Canoe fills at predetermined rate.

During each (speed/4) yard of the trip a single pf jumps in the boat with probability 0.25. He picks a random toe. Cannibals always spear the fish, and half the time make a hole in the boat. Hunters always panic, and capsize the boat. The doctor is quick half the time, and panics half the time. The nurse always panics; half of the time she is calmed down, and the other half, she jumps (alone) out of the boat and must swim ashore.

When the boat capsizes, everybody must swim. Dead players always float to the correct shore, and somehow the canoe gets there, too.

When swimming, the events that follow may occur to each player individually:

Dead players always float ashore.

Live players make it ashore unscathed half the time. The other half, they acquire minor wounds (prob. 0.67) or major wounds (prob 0.33). In no case do they come out of the river healthier than they went into it.

At the present time, these probabilities are independent of the length of the swim. (Improvers take note.)

```
/* Conducts dialog, determining which players make next trip.
whosgoing [
  char j,p,i
  char dup
  pl"";pl"move "
  while(1)[
    j=getchar
    if(j=='.')[    /* Trip command.
      i=0
      while((i=i+1)<=ngoing)      /* At least one paddler required.
        if(health(move(i))<5)return
      ps" nobody to paddle "
    ]
    else if(j=='-')[   /* Unload command.
      ngoing=0
      ps" canoe emptied"; pl""
    ]
    else if(j=='s')[   /* Print board.
      status
      ps"move "
    ]
    else if((j>='1')*(j<='9'))[    /* Put player in canoe.
      p=j-'0'
      dup=0
      i=0
      while((i=i+1)<=ngoing) if(p==move(i)) dup=1
      if(dup) ps" already in boat "
      else if(shore(p)!=canoe) ps" on other shore "
      else if(ngoing>=4) ps" canoe full "
      else move(ngoing=ngoing+1)=p
    ]
  ]
]
```

4-10

4.2.2 Piranha Fish Code

```
char shore(9),health(9),canoe,move(4),ngoing,afloat
int hfactor(6),sinkrate,paddle(9)
/* Conducts the game.
pf [
  setup
  while(stillplaying())[
    whosgoing
    trip
    shoreacts
  ]
  wrapup
]

/* Sets up initial conditions.
setup [
  hfactor(0)=10
  hfactor(1)=9
  hfactor(2)=hfactor(3)=8
  hfactor(4)=7
  paddle(1)=paddle(2)=12
  paddle(3)=paddle(4)=9
  paddle(5)=7
  paddle(6)=5
  paddle(7)=paddle(8)=paddle(9)=4
  sinkrate=25
  ps"seed"
  seed=last=gn
]

/* Game is still going if any player on shore 0 is alive.
stillplaying [
  int p
  while((p=p+1)<=9)
   if((shore(p)==0)*(health(p)<5)) return 1
]
```

```
      break
    ]
    if(afloat)[
      pl"Canoe has"; pn 100-dist; ps" yards to go, and is"
      pn full; ps"% full"
      if(random(1,4)==1)onefish
    ]
  ]
  i=0       /* The far shore is reached.
  while((i=i+1)<=ngoing) shore(move(i))=1-shore(move(i))
  canoe=1-canoe    /* Swap shores of players in canoe, and canoe.
  ngoing=0    /* Everybody out.
  pl"trip to "
  if(canoe)ps"far"; else ps"near"
  ps" shore is complete."
]

/* A fish jumped in the boat. This is what happens.
onefish
[
  char p
  pl"A piranha fish has jumped into the boat.  He is swimming"
  pl"around.  He is looking at the toe of the "
  pname(p=move(random(1,ngoing)))
  ps"."
  if(health(p)>4) pl"Oh, well.  He's dead anyway....."
  else if(p>6)[
    pl"The missionary is calm.  He is staring back at the"
    pl"fish.  The fish just jumped back into the river."
  ]
  else if(p<3)[
    pl"The cannibal has speared the fish.  "
    if(random(0,1))[
      pl"Unfortunately he made a hole in the"
      pl"boat, increasing its sink rate 10%."
      sinkrate=sinkrate+sinkrate/10
    ]
  ]
```

4-12

```
/* status prints the board.
status [
  char k(Ø),p
  pl"";pl""
  ps "near shore                    far shore  "
  pl"";pl""
  while((p=p+1)<=9)[
    if(shore(p)) ps "                                  "
    pn p; ps" "; pname p; ps" "
    if(health(p))[
  k="minor att  major att  minor unattmajor unattdead
      k=k+11*(health(p)-1)
      pft k,k+1Ø
    ]
    pl""
  ]
  if(canoe)ps "                    "
  ps "     canoe"
  pl"    "
  if(canoe)ps "                   "
  char i
  while((i=i+1)<=ngoing)pn move(i)
  pl""
]

/* Conducts a trip across the river.
trip [
  char i
  int speed,dist,full
  afloat=1
  while((i=i+1)<=ngoing)
    speed = speed + paddle(move(i))*hfactor(health(move(i)
  speed=speed/(4*ngoing)    /* Yards per unit of time.
  while((dist=dist+speed)<1ØØ)[
    full=full+sinkrate
    if(afloat*(full>1ØØ))[
      pl"The boat is swamped....."
      capsize
```

```
    if(random(0,2))[
      if(health(p)==2)health(p)=4
      else if(health(p)<2)health(p)=3
    ]
    else if(health(p)<4) health(p)=4
    if(health(p)==3) ps" fortunately escapes with minor wounds"
    else ps " major wounds acquired."
  ]
]

/* The canoe is capsized.
capsize [
  char p
  pl"CAPSIZE!!!  Everybody swim FAST!!  The fish are coming.."
  while((p=p+1)<=ngoing)swim move(p)
  afloat=0
]

/* When on shore, some players get mended.
shoreacts [
  char p
  while((p=p+1)<=9)[
    if(shore(p)==shore(5))[   /* Doctor with at most minor wounds can attend all
      if(health(5)<4) if(health(5)!=2)                    /*    wounds.
        if((health(p)==3)+(health(p)==4)) [
          health(p)=health(p)-2
          pl""; pn p; ps " attended by doctor."
        ]
    ]
    if(shore(p)==shore(6))[
      if(health(6)<4) if(health(6)!=2) if(health(p)==3) [
        health(p)=1
        pl""; pn p; ps" attended by nurse."
      ]
      else if(health(p)==4)   /* (And also major wounds with the doctor's advice.)
        if(shore(5)==shore(6))
          if(health(5)<5) [
```

```
    else if(p<5)[
      pl"The hunter has panicked.  He is rocking the boat..."
      capsize
    ]
    else if(p==5)[
      if(random(0,1))[
        pl"The doctor is quick.  He shoots the fish full of"
        pl"a drug."
      ]
      else[
        pl"The doctor has panicked.  He is rocking the booooooat!"
        capsize
      ]
    ]
    else[
      pl"The nurse has panicked.  She is rocking the boat."
      pl"Everybody is yelling at her.  Yell - yell - yell."
      if(random(0,1))[
        pl"She is calm now, and sits down."
      ]
      else[
        pl"She falls out of the boat.  She is swimming."
        swim 6
      ]
    ]
]

/* Player p swims to shore.
swim char p [
  if(health(p)>4)[
    pl"Player"; pn p; ps" floats ashore."
  ]
  else if(random(0,1))[
    pl"Player"; pn p; ps" makes it."
  ]
  else[
    pl"BYTE!! BYTE!!  Player"; pn p
```

```
/* Prints a player's name.
pname char p [
  char k(Ø)
  if(p<3)ps "cannibal"
  else if(p<5)ps "hunter"
  else if(p<6)ps "doctor"
  else if(p<7)ps "nurse"
  else ps "missionary"
]
```

Comments on Style

Notice how the functionality of this program makes it readable. You can find a feature quickly, and modify it with confidence that the house won't fall in.

Note the use of the character pointer k in status. It is used to compute for printing one of five possible health messages, depending on the health of player p. The function pft is used to print exactly 11 characters. Thus, from three lines of code any one of five messages is printed.

Piranha Fish uses only standard and optional library functions, and standard MCs. These are all furnished with tiny-c. So if you can get tiny-c up, you've got this program in the bag. If you use a plot function from your personal library, dramatic improvements to this game are possible.

```
            health(p)=2
            pl""; pn p; ps" attended by nurse"
          ]
      ]
      if(health(p)==0)[]    /* All done if healthy.
      else if(random(0,2))[]    /* All done for .67 of sick.
      else if(health(p)<3)[     /* But some get sicker.
        if(random(0,2)==0)[
          if((health(p)=health(p)+1)==3) health(p)=5
          pl""; pn p; ps" is much worse"
          if(health(p)==5) ps", in fact dead."
        ]
      ]
      else if(health(p)<5)[
        health(p)=health(p)+1
        pl""; pn p; ps" is much worse"
        if(health(p)==5)ps", in fact dead."
      ]
    ]
  ]

  /* Computes score.
  wrapup [
    int s,h,p
    s=1000      /* Perfect score.
    while((p=p+1)<=9)[
      h=health(p)
      if(h==5)s=s-100
      if(h==4)s=s-30
      if(h==3)s=s-15
      if(h==2)s=s-10
      if(h==1)s=s-5
    ]
    pl"";pl""
    status
    ps"Your score is"; pn s
  ]
```

ASSEMBLER INPUT-OUTPUT ROUTINES

for the MZ-80K

A BASIC programmer is heavily insulated from the raw machine that lurks beneath his fingers. Indeed, it is largely the function of any high level language to hide all the low level details and present a clean, unified, interface to the user in the form of suitable control structures, arithmetic and functions that are meaningful and easy to use. The price fo such elegance is high, particularly with a BASIC Interpreter, and takes the form of a spectacular drop in performance. A BASIC program may be as much as 100 times slower than the equivalent assembler code program. The converse is also true and the assembler programmer must pay for the speed advantage by spending perhaps 5 to 10 times the amount of time and effort developing his program. This presents the programmer with a dilemma to which two of the classic answers are:-

1. Use a high level language which compiles efficient machine code i.e. Optimised FORTRAN, CORAL 66, etc. obviously a suitable language must be available for this solution to be adopted.

2. Use a generalised assembler package developed by someone else.

This article addresses the second solution and is concerned with a package of Input-Output Subroutines for the MZ-80K written by Bateman Software.

The MZ-80K ROM contains a very limited number of useful Input-Output routines to handle the visual display unit (VDU) and the cassette unit. The printer and disc drives are totally ignored. Since the user with discs will have had an operating system supplied which will contain many useful Input-Output routines, no attempt has been made to provide them. This leaves the VDU, printer and cassette unit.

To be useful, any generalised Input-Output package should adhere to certain rules, These are:

a. The different devices should, as far as possible, be treated in the same way and use similar names for similar functions. This helps the programmer remember the function and interface for the routines and therefore reduces the possibility of errors.

b. The more powerful routines should be based upon low level routines whenever possible. In the limit, a small number of single character Input-Output routines should form the basis of the whole package. This is a significant aid to portability since only the basic routines will need to be changed in order to transfer the program to another machine.

c. The routines should handle the standard types of data that an assembler programmer is most concerned with e.g. single characters, character strings and integers. (Floating Point numbers are likely to be too specialised for such a general package and are therefore omitted in order to reduce the size of the package. They may be added very easily if required.)

d. The routines should be well defined and safe to use so that interesting side effects, e.g. clearing the store, do not occur. To this end all registers should be stacked whenever possible and the registers should be used as consistently as possible throughout the package.

Having settled the ground rules I can now describe the implementation. The VDU screen and the printer can both be regarded as simple, single character at a time, devices and thus be treated identically. Obviously, the printer has no equivalent of the VDU's keyboard, which is also a simple, single character at a time device. It would seem reasonable to try to treat the cassette unit as a single character Input-Output device so that it can be handled in the same way as the VDU and printer. The programmer could then simply switch Input-Output between all the devices without having to modify it in any way. Unfortunately, the mechanics of the cassette unit tend to inhibit a direct implementation of this because the tape is stopped and started for every transfer, however big or small it may be. In the case of single character transfers the inter-record gap (the stop-start takes a significant time and amount of tape that cannot be used) would be considerably larger than the data itself. Very little data could be stored on a C12 tape. To overcome this problem the single character transfers are buffered in store i.e. a number of bytes of store, say 128, is set aside to receive the characters and, when the buffer is full (for writing to tape) or entry (for reading from tape) the total buffer is transfered. The size of the buffer is obviously a compromise which depends upon the amount of tape that may be wasted in inter-records gaps and the amount of store than can be set aside for the buffers. The package to be described allows the user to select his own buffer size and position. One further problem arises with tape that does not occur with the VDU or printer. Since only complete bufers of data may be transferred then some method must be found for completing transfers at the end of the file because it is very unlikely that the buffer just happens to be full at this point. The method chosen is to simply output a special character, called End Of File, which is stored in the buffer and then forces a tape transfer. On reading, the EOF character will appear as a normal character value following a read and the program must detect it and stop transferring data and close the file.

I can now describe the actual subroutines supplied in the package. These will be listed in terms of increasing power and will be grouped by function whenever this applies to more than one device. The devices are denoted by a 'V', for VDU, 'P', for printer, and a 'T', for the tape, following the basic function name.

The subroutines are:-

OUTV OUTP OUTT	Outputs one character to the device
NLV NLP NLT	Outputs a newline to the device
SPV SPP SPT	Outputs a space to the device
EOF	Outputs an 'End Of File' to the tape unit
COPYV COPYP COPYT	Outputs a string of characters to the device
LINEV LINEP LINET	Outputs a new line following by a string of characters to the device.
PRINTV PRINTP PRINTT	Prints a 16 bit signed integer in decimal or Hex. format to the device
INKEY INTAPE	Inputs a characters from the device
STRINGKEY STRINGTAPE	Inputs a string of characters from the device
READKEY READTAPE	Inputs a decimal or Hex. integer from the device
OPENR OPENW	Opens a named file for reading or writing
READFILE WRITEFILE	Reads or writes a complete file (as opposed to a data file using OUTT, PRINTT, etc.) to or from tape
VERIFYFILE	Checks the file on tape agains the copy in store
READOBJ WRITEOBJ	Reads or writes an object file. The user is prompted for the start finish, load and execute addresses.
VERIFYOBJ	Verifies the object file on tape against the copy in store.
DEBUG	Prints the contents of all the registers in Hex. on the printer it may me modified to use the VDU by changing the appropriate calls inside DEBUG and by inserting a CALL READKEY to prevent the data from scrolling off the screen inadvertently.

The package also contains a small test program intended to illustrate the method of using the subroutines and also to indicate the error trap for detecting run-time errors e.g. typing a non-Hex. digit - 4EJ6H.

The cassette supplied contains both the source and object files and the manual contains a source listing as well as a detailed description of the subroutines. This means that, although written for a ZEN assembler, no difficulty should be experienced in adapting it to any other assembler. The complete package requires 1408 bytes of store and it may, of course, be cut down to your own requirements.

R.F. BATEMAN

BATEMAN'S SOFTWARE IS AVAILABLE FROM SHARPSOFT PRICE £14.00

```
**********************
**********************
**                  **
**  L E T T E R S   **
**                  **
**********************
**********************
```

Dear Sirs,

The following might be of interest for inclusion in the User Notes under the Machine Code section.

1. When saving a memory block onto tape using the calls to the locations 0021 (to save the file name) and 0024 (to save the block of data). The file name is stored from locations 10F1 H to 10FF H in ASCII format e.g. SHARPSOFT appears as:-

10F1 53 48 41 52 50 53 4F 46 54 (all Hex

2. Cells 1104H and 1105H contain the start address of the block in Lo Hi format e.g. 1234H appears as:-

cell 1104H=34H
1105H=12H

3. Cells 1102H and 1103H contain the NUMBER OF CELLS TO BE SAVED NOT the end address. e.g. to save the block from 1000H to 2000H

cells 1102H to 1105H would contain:-

00 10	00 10
No. of cells	Start Address

4. When loading the tape file name, the computer puts it in cells 10F1H to 10FFH (see 1).

5. When loading or verifying a block of data the user cannot alter where it is loaded into as this is set on saving (see 2 and 3).

Can any reader supply a machine code subroutine which will switch a printer on or off so that everything which is printed on the screen is also sent to be printed on a P3 printer, if this is possible. If it is not possible can anyone simplify the machine code listing in the P3 manual as I cannot make any sense of it as there are no useful comments.

W. HOWARD

****

Dear Sharpsoft,

As requested, and for anyone interested in User Notes, to use "Print Using" in Double Precision Basic (Disc) SP-6115 with # (pound sign!) in formatted output in lieu of @, then Poke 20867,251 or in lieu of $, then Poke 20865,251. I think a bonus for serious business users.

Regarding User Notes Issue 3, Mr. Sharman's letter (I think - typo error - anyway the one starting on page 57), his Pokes to convert the clock function, even in the order printed, do not work. Perhaps he could scream if his tip was misquoted - I should like to know.

User Notes assuredly good value, even at the new price.

N. DIGHT
LONDON

****

Dear Sir,

Your 3rd Newsletter was a welcome sight after the long gap. I found it surprising that you should have used the same line numbers for each of the BASIC utility programs so that only one could be loaded at a time without rewriting - not very well planned. The poor proof reading was also very disappointing.

Moans aside, the contents were very interesting and some of the hints e.g. how to make the clock run fast, have already been useful. I would be interested to see articles on details of useful routines in the SP-6015 Disc BASIC - little seems to be published in comparison with the hints for SP-5025. Some more general ideas on tricks with the 8255 and 8253 chips would also be of interest.

I enclose my cheque for #7.50 for next year's subscription to the User Notes and would also be grateful for fuller details of the Speed BASIC and whether or not the cassette speed can cope with 4Mhz since I have fitted a Z80A to my MZ-80K.

D.H. JACKSON
YORKSHIRE

****

Dear Sirs,

Please find enclosed a short music program for the MZ-80K which may be of interest to your readers of the User Notes.

I have computerised a short tune which may or may not be subject to copyright. I will leave it to you to deal with it as you see fit. You may do anything you wish with it.

R.A. HILL
SHEFFIELD

```
10 REM COMPUTERISED MUSIC JINGLE BY R.A.HILL
20 TEMPO6
30 C$="^C2B^D^C":D$="^C2R1^C4R1^C2^D4^#D^E6R4 "
40 E$="^C6R4":F$="A2GFEGFEDE4#D2E4#D2E6R2"
50 G$="^D1R^D2^#C^E^D#A2"
60 H$="G1RG2#FAGFEF4E2F4E2F6R2"
70 J$="G1RG4#G2A2R2A2G4#G2A2RAG4^F2^E4^D2^C4R1"
80 MUSICC$,"R2",C$,"R2",D$
100 MUSICC$,F$
110 MUSICG$,H$
120 MUSICC$,F$
130 MUSICJ$
140 FORK=200TO100STEP-10 :FORJ=1TO50STEP15 :USR(68):POKE4515,J:POKE4514,K
150 NEXTJ,K:POKE4514,10:USR(71)
160 FORD=1TO1000:NEXT
200 GOTO10
```

****

Dear Sir,

Please could you tell me if the Xtal Serial I/O Interface Card can be used with Sharp Cassette Basic and/or Disc Basic.

Could you also tell me if it is possible to transfer existing Basic and Machine Code programs (cassette) onto Disc, and if so would any POKE commands need altering if they refer to the cassette Basic.

D. JERVIS
MIDDX.

As far as we are aware the existing printer drivers included in SHARP BASIC would not drive the XTAL Serial I/O interface card. If you are a machine code "hacker" then, with some work, it should be possible to patch the BASIC code.

SHARP Disc BASIC includes a utility for transferring programs on cassette to disc - this will transfer both BASIC and machine code programs.

EDITOR

****

Dear Sirs,

Towards the end of last year I purchased from you a Sharp MZ-80K, together with an Epson MX80-FT printer. Shortly afterwards I received from you the first three 1981 issues of Sharpsoft User Notes and I am eagerly looking forward to receiving the first edition of 1982.

I have had no prior experience of computing or of programming, although in the few months before ordering the computer from you I did read up a little in books and magazines - frankly, hardly understanding much of what I had read.

The most surprising thing was that within a very few hours of getting delivery of the computer so much of what I had read (and not understood!) suddenly fell into place. What is abundantly clear is that if one wishes to get to grips with computing, with programming and BASIC language there is nothing like "hands on" experience. Most definitely I would strongly recommend to anybody who wishes to come to terms with computing and programming that they should get themselves a computer at the earliest possible opportunity. I simply do not believe that one can understand what it is all about without actual practical experience - even if that experience is in ones own home without any tutor or instructor.

I am truly amazed at the progress I have managed to make in the few months since I received my equipment, working alone and with nothing other than a couple of books and the User Notes.

Still, I do have problems that I have not managed to resolve. Perhaps it is my lack of theoretical background to computing that causes the difficulty but I find that I simply cannot understand how to build into my programmes some of the printer instruction commands. Is it possible that a sheet could be prepared and circulated giving these instructions in simple everyday terms? For example, where the handbook and the notes that you supplied with it tell me to enter various hexadecimal values. Pardon my ignorance, but my interpretation of the term "hexadecimal" is in the semi-Churchillian phrase of "cursed dots"! Any assistance you can provide in explaining precisely what instructions have to be entered will be of great assistance.

Another problem I have is that my work is in the financial field and programmes I have developed for my own professional use do require tabulation of columns of figures. Both on the screen and on the printer these come out with the figures justified to the left i.e. lining up the highest values of each number) instead of being justified to the right (i.e. lining up the units inch figure under each other). How does one overcome this problem?

Turning to issue three of the User Notes, I saw with great delight that they included listings of utility programmes and I felt that I could make immediate use of the utility programme "renumber". I laboriously typed it into the machine and then saved it on tape. With great expectations I then loaded in a rather lengthy programme I had developed and which I wished to renumber. The result? No help at all!

Am I correct in assuming that whenever the utility programmes mentioned in the User Notes are used, the programme to which they are to be applied has to be typed into the computer and not loaded from cassette, having previously been saved?

Incidentally, in the User Notes it mentions that "copies of notes". Unfortunately I received only the notes, not a cassette in sight!

In case this letter seems to you to be a tale of criticism and complaint, let me add strength to that by telling you that in my wife's opinion your services are sorely defficient in another way. In her view there should be a law that no computer equipment is ever advertised, displayed or sold without a notice being prominently shown to the effect that "**Computing can seriously damage your marriage**"!

Seriously, I have found the whole subject of computing and programming to be absolutely fascinating and I think that my "complaints" should be taken more as an expression of my eagerness to learn more and more about it.

Finally might I add that since getting delivery of the computer it has become a little difficult to trace my youngest son (aged 10) and also it has often been difficult for me to get to work on the computer until the wee small hours of the morning. The two problems are not unconnected since if we want Stephen for anything, the first place to check, if you want to find him, is the computer!

M.S. ZATMAN F.C.A.
MIDDLESEX

To print numeric values justified on the right use the following subroutine:-

```
X=.001
IF V<0 THEN X=-.001
V=V+X
V$=STR$(V)
VL=LEN(V$)
VL=VL-1
V$=LEFT$(V$,VL)
RETURN
```

To use the subroutine put the value to be printed in V. GOSUB to the subroutine. On return the value is now in V$, notice it is now alpha.

To right justify this value use the following:-

```
PRINT TAB(Y-VL);V$
```

where Y is the rightmost column.

For printer output use:-

```
PRINT/P TAB(Y-VL);V$
```

Control codes for the Epson:

```
                HOME = PRINT/P CHR$(12)
   LARGE CHARACTERS = PRINT/P CHR$(14)
CONDENSED CHARACTERS = PRINT/P CHR$(15)
               CLEAR = PRINT/P CHR$(20);CHR$(18)
```

EDITOR

****

Dear Sharpsoft,

Thank you for printing my letter in your magazine. Here are a few more useful tips.....

To disable the BREAK key enter at the start of the program:
POKE 6636,0:POKE 8767,0:POKE 8768,133:POKE 8769,19

To re-enable, enter:
POKE 6636,205:POKE 8767,218:POKE 8768,133:POKE 8769,19

If you do not have a PRINT AT, your can do something similar by entering the following short program, running it, and then you can make a copy of BASIC with it included by executing USR calls 33 and 36.

```
100 DATA 205,139,22,91,28,205,169
110 DATA 25,123,254,25,210,152,19,50
120 DATA 114,17,205,154,22,44,205,169
130 DATA 25,123,254,40,210,152,19,50
140 DATA 113,17,205,154,22,93,195,69,28
150 FOR I=15836 TO 15876:READ D:POKE I,D:NEXT I
160 POKE 7221,220:POKE 7222,61
170 POKE 4354,5:POKE 4355,44
180 POKE 4350,65:POKE 4351,13
```

To use the above, the PRINT statement is formatted as follows:

```
PRINT [X,]"HELLO!"
        /
   the coordinates
```

PIERS HENDRIE
CB3 OEG

****

Dear Sirs,

Here's a useful tip that prevents 'SNOW' from coming onto the screen when poking graphics onto it. To illustrate this SNOW, type in and RUN the following program:-

```
10 PRINT"CLS"(CLEAR SCREEN):FOR A=53248 TO 54247:POKE A,200:POKE
A,0:NEXT
```

Now edit the program to the following:-

```
10 PRINT"CLS":B=3494:FOR A=53248 TO 54247:USR(B):POKE A,200:USR(B):
POKEA,0:NEXT
```

Run it. This slows the speed down considerably but the display looks much nicer. The USR call 3494 stops the snow, and it should be inserted before any screen poke.

J. PORTERFIELD
LONDON

Dear Sharpsoft,

Please find enclosed subject to '82 User Notes. I found the '81 Notes very interesting and extremely helpful. Re: Parker's suggestion in note No.3 (page 58) of 'cursor addressing' as described in 'Practical Computing' Nov. '80 (page 109). I located the ish. and attempted operation which was totally unsuccessful. I would appreciate some advise on this matter if possible and have re-printed listing and instructions for Cursor Addressing, have Sharp altered the MZ-80K or the BASIC SP-5025 so as to render this program redundant?

- Load BASIC SP-5025 - key program & check - insert blank tape
- Key RUN - Displays - record - play & WRITING BASIC SP-5025A
- Final check - Displays - FINISHED at centre of screen.

```
 1 DATA 205,139,22,91,69,28,205,169
 2 DATA 25,123,254,25,210,152,19,50
 3 DATA 114,17,205,154,22,44,205,169
 4 DATA 25,123,254,40,210,152,19,50
 5 DATA 113,17,205,154,22,93,195,69,28
 6 FOR I=15836 TO 15876:READ E:POKEI,E:NEXT
 7 POKE 7221,220:POKE 7222,61
 8 POKE 4354,5:POKE 4355,44
 9 POKE 4350,65:POKE 4351,13
10 USR(33):USR(36)
11 PRINT"C":PRINT(10,16)"FINISHED"
```

Thanking you in anticipation.

MAEEKELBERGHE-JACKSON

****

Dear Sirs,

As a newcomer to computing I found the first two issues of the SHARPSOFT User Notes very interesting. I offer the following comments, for what they're worth, based on what I've discovered by playing around with my MZ-80K.

RUN — While it is true that RUN zeros all variables, it is not true that RUN followed by a line number does the same. Try the following simple program:

```
10 PRINT A
20 END
```

Type in A=1 as a direct command and then RUN. As you would anticipate a zero is printed. However, if you repeat this with RUN replaced by RUN 10 the number 1 is printed.

CLR — This does more than zero all variables, it eliminates them altogether. Thus, if CLR appears after a DIM statement in a program you will find that any subscripted variables you thought you had dimensioned don't in fact exist any more! Incidentally, CLR doesn't have the same effect as RESTORE on the READ/DATA pointer (as suggested in User Notes issue 1).

RIGHT$ This is defined incorrectly on page 17 of User Notes issue 1. It returns a string of characters equal to the rightmost N characters, not equal to the rightmost characters counted from the Nth position. Coincidentally, the example given on page 17 is correct for both definitions!

OR Both the SHARP manual and your User Notes mention that the + sign can be used to represent a logical OR. What neither mentions is that this is an Inclusive OR - that is, it means either the one or the other or both. One can produce a logical Exclusive OR - that is, either the one or the other but not both - simply by replacing the plus with a minus sign.

When is equality not equality? I tried the following simple program with surprising results (at least, they surprised me, though presumably they are well known to many):

```
10 FOR K=0 TO 1STEP .125
20 PRINT K
30 IF K=.125 THEN 60
40 NEXT K
50 GOTO 70
60 PRINT "LINE 60"
70 END
```

As expected, this program results in:

```
0
.125
LINE 60
```

However, if line 30 is changed slightly to:

```
30 IF K=.25 THEN 60
```

the following unexpected sequence results:

```
0
.125
.25
.375
.5
.625
.75
.875
1
```

In other words the mathematical equality on the third loop is not recognised! It is similarly not recognised if the .25 is replaced by .375, .625, .75 and .875. On the other hand it is recognised for .5 and 1. In general, there seems to be an enormous number of such non-integer equalities which are not recognised as such. I chose .125 in the above example because I originally thought the problem might not apply to (negative) integer powers of 2 (.125=2 -3). As is seen I was wrong. Does anyone know how to predict in advance the conditions for which this problem will occur? There are several ways of overcoming this problem in practice, the simplest I have found being illustrated below for the example given above; namely:

```
30 IF K-.25=0 THEN 60
```

This works where the original doesn't!

Finally: Is there a simple POKE routine which would produce a RESTORE "line number" for the tape version of SHARP BASIC?

Is there any way of using the internal clock of the MZ-80K to time fractions of a second?

How can one change the volume control on the sound?

What is the difference between Assembler and Machine language?

I look forward to future issues of the User Notes. I hope you manage to produce many more issues next year.

ALAN STEVENS
DERBY

The volume control for sound is inside the machine and can be accessed by opening the MZ-80K. The control is clearly labelled VOLUME.

An Assembler ultimately produces Machine Code programs but is far easier to use giving the programmer additional facilities.

EDITOR

****

Dear Sir,

Please find enclosed a printout of a programme that I wrote on our MZ-80K. It is called Maze Raze and is quite short and easy to type in and should be quite fun for newcomers to computers.

I have been interested in programming for about six months and have found your User Notes very interesting and helpful. We have recently bought a disk unit and an Epson printer and I am finding disk basic a pleasure to use after waiting for cassettes to load.

R.YOUNG (aged 10)
MIDDLESEX.

MAZE RAZE

A timed race around the Maze, going clockwise - no cheating now!

Control keys are Q W E, A D, Z X C.

There are varying speed rates FAST or SLOW (1-7)

This programme will run on SP-5025 or SP-6015.

```
10 REM Program by R.YOUNG.
20 REM 7/1/82
30 GOSUB560
40 PRINT"■"
50 PRINT"    ┌──────────────────────────┐"
60 PRINT"    │              ▒           │"
70 PRINT"    │              ▒           │"
80 PRINT"    │    ┌─────────────────┐   │"
90 PRINT"    │    │                 │   │"
100 PRINT"    │    │   ┌─────────┐   │   │"
110 PRINT"    │    │   │         │   │   │"
120 PRINT"    │    └───┼─┐       │   │   │"
130 PRINT"    │        │ │       │   │   │"
140 PRINT"    ├────────┘ │       │   │   │"
150 PRINT"    │          │       │   └───┘"
160 PRINT"    │   ┌──────┘       │       │"
170 PRINT"    │   └──┐           │       │"
180 PRINT"    │      │           └──┐    │"
190 PRINT"    └──┐   │              │    │"
200 PRINT"       │   └──────────────┘    │"
210 PRINT"       │                  │    │"
220 PRINT"       │      │           │    │"
230 PRINT"       │      │           │    │"
240 PRINT"       └──────┴───────────┴────┘"
250 PRINT"SPEED ";FS$;:IFFS$="FAST"THEN270
260 PRINTT
270 M=100
280 A=53248
290 TI$="000000"
300 GETD$:IFD$=""THEN320
310 GETA$
320 IFD$="Q"THENC=-41
330 IFD$="W"THENC=-40
340 IFD$="E"THENC=-39
350 IFD$="A"THENC=-1
360 IFD$="D"THENC=1
370 IFD$="Z"THENC=39
380 IFD$="X"THENC=40
390 IFD$="C"THENC=41
400 IFO=1THEN420
410 TEMPOT:MUSIC"RO"
420 IFPEEK(A+M+C)=209THEN530
430 IFPEEK(A+M+C)=0THEN490
440 POKE4514,80
450 FORI=1TO5STEP.1:POKE4513,255:USR(68):NEXTI
460 MUSIC"RO"
470 D$="S"
480 M=M-C:GOTO490
490 M=M+C
500 POKEA+M,199
510 POKEA+M-C,0
520 GOTO300
530 PRINT"■ YOU PASSED THE LINE IN";VAL(TI$);" SECS"
540 GETA$:IFA$=""THEN540
550 RUN
560 PRINT"■■■ SELECT YOUR SPEED"
570 PRINT" FAST OR SLOW ";
580 INPUT FS$
590 IFFS$="FAST"THENO=1
600 IFFS$="SLOW"THENINPUT" 1-7 ? ";T
610 IFFS$="SLOW"THENTEMPOT
620 RETURN
```

Dear SUN, (Sharpsoft User Notes)

In reply to the query from Mr. D. McD. Wood published on page 61 of Issue 3 the short BASIC programme enclosed will allow any number of previously saved SP-5025 programmes to be Appended together subject to the usual conditions (memory size and non-overlapping line numbers).

The programme should be loaded immediately after BASIC and thereafter the facility to Append will be offered each time RUN is entered. If the facility is not needed CONT will cause programmes to RUN as normal.

Alternatively the programme may be typed in at any ti me required: it is short enough to make this quite practicable: only remember to leave lines 1-9 free for this purpose (which most programmers do anyway).

Lines 1-9 may be left in a programme whilst it is under development at very little inconvenience but would normally be deleted before saving the final version to tape.

For those who are interested, the programme works by changing the relevant pointers in SP-5025 to 1 above the end of the BASIC text currently in memory, rather than 4806H, when LOADing.

APPEND Copyright M.A. Hawes 1980

```
1 PRINT "CLEAR SCREEN":POKE10167,1:AP=PEEK(17972)-2:AQ=PEEK(17973)
2 IFAP<OTHENAP=AP+256:AQ=AQ-1
3 FORN=ITO5:READLP,LQ:POKELP,AP:POKELQ,AQ:NEXTN
4 DATA1097,10948,10972,10973,10984,10985,11081,11082,11105,1106
5 PRINT"ON'READY'LOAD FIRST/NEXT PROGRAMME IN USUAL WAY THEN TYPE 'RUN'
6 PRINT:PRINT"IF NOT APPENDING TYPE 'CONT' THIS WILL RUN MERGED PROGRAMMI
7 PRINT:STOP
8 AP=6:AQ=72:FORN=ITO5:READLP,LQ:POKELP,AP:POKELQ,AQ:NEXTN
9 DATA1097,10948,10972,10973,10984,10985,11081,11082,11105,1106
```

M.A. HAWES
SHROPSHIRE

****

Dear Sharpsoft,

Congratulations on your excellent Third Issue, keep up the good work. It's nice to see that the MZ-80K is now benefiting from its original design as a 'Clean Computer' in addition to BASIC, I now have PASCAL (SHARP) and PILOT, with FORTH soon to come.

In Issue 3 reference was made to a 'Micro-Pilot'. Perhaps your readers would be interested to know of 'Wirral Pilot'. This is an extensive integer version avaiable from Mr. Alec Wood, Wirral Grammar School, Cross Lane, Bebington Wirral, L63 3AQ. The language tape plus a full definition of the Pilot language cost me ten pounds last June, anyone interested should contact the School.

Now an appeal for HELP. Despite numerous errors the SHARP PASCAL Manual is comprehensive, however it has several serious omissions. Can anyone help with any of the following?

1. Routine to drive the internal Clock?

2. Routines to generate common sounds (sirens, alarms etc).

3. A routine to blank the screen.

P. BRAITHWAITE
CULLERCOATS

****

Dear Sharpsoft,

BULLET PROOF is a subroutine which I think makes a program impossible to run, list or whatever until a present keyword is typed in. Be warned that failure to enter the correct keyword will erase both the program and the BASIC.

Congratulations on your User Notes which are truly a mine of information.

Some more Pokes etc.

POKE 18440,1 and POKE 18441,0 Restores Line 0 to Line 1 so it can be tampered with.

POKE 10682,1 allows a program to RUN from LOAD.

Many thanks for a wonderful service.

K. OLLETT
EAST SUSSEX

(BULLET PROOF listing at end of User Notes)

****

Dear Sharpsoft,

I have a problem that maybe you or one of your members may be able to help me with. I have an Epson MX80 printer and I use the BASIC Extensions as designed by Dr. Gladman. If I apply the printer modifying cassette to normal SP-5025 BASIC everything is O.K. but when I try and use the BASIC SP-5025 plus the BASIC Extensions plus the printer modifying tape I find that the printer will only print capitals and won't respond to the special commands e.g. double width etc. Is there a way of correcting this?

As far as I know I have the only MZ-80K in New Zealand, do you know of anyone else with one. If so I would like to meet up with them to exchange ideas etc.

J. BEGG
NEW ZEALAND

As far as using your Epson with BASIC Extensions, I am unable to help. However, using Speed Basic with the MX80 overlay does work and works well. If any reader has found a way around the problem occurring with BASIC Extensions/MX80 overlay please let me us know and we will pass the information on to Mr. Begg as well as publishing it in Issue 6.

EDITOR

Dear Sirs,

Your User Notes are interesting - so much so that I await each successive issue with bated breath and my major criticism is that they do not come out often enough.

I would like to make three small contributions:

1.
```
10 GET Z$:IF Z$ = " " GOTO 10.
20 IF Z$ = "A" GOTO 100
25 IF Z$ = "B" GOTO 200
 .
 .
 .
70 IF Z$ = "J" GOTO 1000
75 GOTO 10
```

This menu selection system is frequently used and is long and cumbersome. No doubt many people use the alternative:

```
10 GET Z$:IF Z$ = " " GOTO 10
20 ON ASC(Z$) -64 GOTO 100,200,300,400,500,600,700,800,900,1000
30 GOTO 10
```

2. I have been using Modified BASIC (with Speed Data) exclusively since last August and I find it is so vastly superior to BASIC SP-5025 that I wonder why it has not been adopted as standard.

There is however one peculiarity I have found which will give rise to a SYNTAX ERROR even after very many successful runs of a program. Invariably this is traced to PRINT@ without a space. Changing to PRINT @ will clear the error situation. The one additional feature I would like to see is ON ERROR GOTO....

Cassette DATA recording with Speed BASIC is a very welcome improvement - particularly for those with large amounts of data and little money for disks. I find the 3X claim correct and is in fact about twice as fast as the MZ-80B in this regard.

3. On the MZ-80K with SP-5025, Modified BASIC or Speed BASIC I am unable to get PRINT CHR$ (44) to print the comma (,).

I hope these comments will assist some readers.

R.N. HUMPHREYS
MID. GLAMORGAN

****

Dear Sirs,

In the "Exercises" part of "An Introduction to Z80 Assembly Code Programming" (User Notes 4) you asked the readers to better the execution time of EXAMPLE2.

Please find enclosed my version which uses 5004302 T-states (2.502151 sec. at 2MHz) whereas the routine presented in the User Notes appears to need 5258513 T-states (2.6292565 sec.).

It wasn't too hard to do because the author of EXAMPLE2 apparently go the MZ-80K video RAM size wrong. As we all know the screen is only 25 lines at 40 characters large, i.e. 1000 locations. This is 24 bytes less than 1K, although Sharp wastes 4K by ambiguous decoding of the D-area.

After we filled half the screen only one pointer (HL) needs repositioning, the other (DE) is already where we want it to point at. Now, if we use the B register as the counter, we can facilitate one of the Z80 specials, namely DJNZ.

The unconditional jump at the end of the programme is to the SP-2001 MACHINE LANGUAGE warmstart.

For memory misers (I am one myself), I enclose another version of our graphics job. This is somewhat slower (5387790 T-states) but needs less storage space.

Re the FORTH introduction the demo in tutorial 6 clan be changed to :

```
SCREEN-DISPLAY    HEX
                  FF 00 DO
                        D000 3E8 I FILL
                     LOOP
                  DECIMAL ;
```

this will cut the execution time down to 7 seconds. (As mentioned above, the screen size is only 1000 bytes!)

I don't know which BASIC interpreter the author used for his comparisons, but on my 2 MHz MZ-80K the programme needs "only" 12 minutes and 47 seconds. We can time this easily by extending the programme by 2 lines:

```
 5 TI$="000000"
55 PRINT "C";TI$
```

E. RAMM
GERMANY

Faster Version Of EXAMPLE2

```
2000   AF        START:  XOR  A              ; CLEAR ACC.                          4
   1   2100D0    OUTER:  LD   HL,D000H       ; VIDEO RAM START                    10
   4   11FAD0            LD   DE,D0FAH       ; PUT DE 1/4 SCREEN AHEAD OF HL      10
   7   06FA              LD   B,FAH          ; 1/4 SCREEN                          7
   9   77        LOOP1:  LD   (HL),A         ; WRITE A INTO FIRST QUARTER          7
   A   12                LD   (DE),A         ; WRITE A INTO 2nd   QUARTER          7
   B   23                INC  HL             ; UPDATE POINTER                      6     9745
   C   13                INC  DE             ; UPDATE POINTER                      6
   D   10FA              DJNZ LOOP1          ; DO 250 TIMES                  (8)  13
   F   21EED2            LD   HL,D2EEH       ; PUT HL 1/4 SCREEN AHEAD OF DE      10
  12   06FA              LD   B,FAH          ; 1/4 SCREEN                          7
  14   77        LOOP2:  LD   (HL),A         ; WRITE A INTO 4th QUARTER            7
  15   12                LD   (DE),A         ; WRITE A INTO 3rd QUARTER            7
  16   23                INC  HL             ; UPDATE POINTER                      6     9745
  17   13                INC  DE             ; UPDATE POINTER                      6
  18   10FA              DJNZ LOOP2          ; DO 250 TIMES                  (8)  13
  1A   3C                INC  A              ; NEXT DATA                           4
  1B   C20120           JP   NZ,OUTER       ; DO 00-FF                           10
  1E   C36012           JP   1260H          ; SP-2001 WARMSTART                  10
```

Short Version Of Graphics Job

5004302 T-states
≙ 2.502151 sec.

```
2000   AF        START:  XOR  A              ;CLEAR ACC.                           4
   1   01E803    LOOP:   LD   BC,03E8H       ;LOAD BYTE COUNTER WITH SCREENSIZE   10
   4   2100D0            LD   HL,D000H       ;SOURCE POINTER                      10
   7   11D01D0           LD   DE,D001H       ;DESTINATION POINTER                 10
   A   77                LD   (HL),A         ;PRIME FIRST LOCATION                 7
   B   EDB0              LDIR                ;FILL SCREEN                    (16) 21
   D   3C                INC  A              ;NEXT DATA                            4
   E   C20120           JP   NZ,LOOP         ;DO 00-FF                            10
  11   C36012           JP   1260H           ;SP-2001 WARMSTART                   10
```

5387790 T-states
≙ 2.693895 sec.

Dear Sirs,

As Sharp Pascal lacks some of the features of BASIC I have been attempting to write short routines to fill the gap. I enclose two routines:

First a function to allow the user to calculate powers. This is called by the statement POWER (X,Y)

e.g. NUMBER: = POWER (X,Y)

This is equivalent to calculating X Y in BASIC.

```
FUNCTION POWER(X,Y:REAL):REAL;
VAR Z:INTEGER;P:REAL;
BEGIN
  IF X>0.0 THEN POWER:=EXP(Y*LN(X))
  ELSE IF Y=0.0 THEN POWER:=X/X
  ELSE IF(Y=0.0)AND(Y>0.0)THEN POWER:=0.0
   ELSE IF Y-FLOAT(TRUNC(Y))=0.0 THEN
BEGIN
IF Y>0.0 THEN
  BEGIN P:=1.0;
   FOR Z:=1 TO TRUNC(Y)DO
  P:=P*X
END
  ELSE
    BEGIN
      FOR Z:=1 TO-TRUNC(Y)DO
      P:=P/X
    END;
    POWER:=P
  END
 ELSE POWER:=SQRT(-1.0)
END;
```

The second routine performs the SET and the third routine performs the RESET. These are a little slow but do the job. Could anyone speed them up?

```
PROCEDURE SET(X,Y:INTEGER);
  VAR A,B,PP,OL:INTEGER;
      XB,YB:BOOLEAN;
  BEGIN
    XB:=ODD(X);YB:=ODD(Y);
    IF XB THEN A:=2 ELSE A:=1;
    IF YB THEN B:=4 ELSE B:=1;
    PP:=-12288+X DIV 2+40*(Y DIV 2);
    OL:=ORD(PEEK(PP))-240;
    IF OL<0 THEN OL:=0;
    IF OL DIV(A*B)MOD 2=0 THEN
    POKE(CHR(A*B+240+OL),PP)
  END;
PROCEDURE RESET(X,Y:INTEGER);
  VAR A,B,PP,OL:INTEGER;
      XB,YB:BOOLEAN;
  BEGIN
    XB:=ODD(X);YB:=ODD(Y);
    IF XB THEN A:=2 ELSE A:=1;
    IF YB THEN B:=4 ELSE B:=1;
    PP:=-12288+X DIV 2+40*(Y DIV 2);
    OL:=ORD(PEEK(PP))-240;
    IF OL<0 THEN POKE(CHR(0),PP)
    ELSE IF OL DIV(A*B)MOD 2=1 THEN
    POKE(CHR(240+OL-A*B),PP)
END;
```

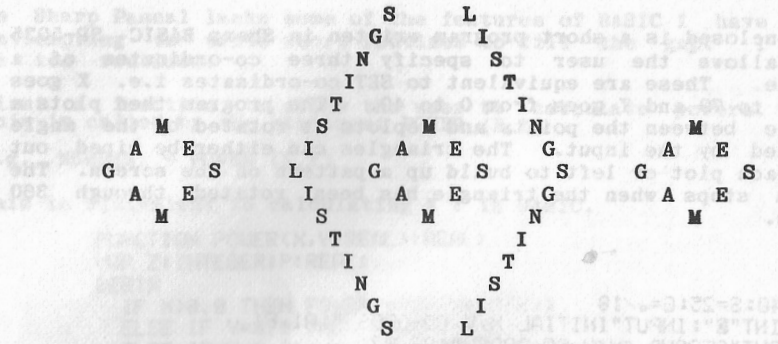
Also enclosed is a short program written in Sharp BASIC SP-5025 which allows the user to specify three co-ordinates of a triangle. These are equivalent to SET co-ordinates i.e. X goes from 0 to 79 and Y goes from 0 to 49. The program then plots a triangle between the points and replots it rotated by the angle specified by the input. The triangles can either be wiped out after each plot or left to build up a pattern on the screen. The program stops when the triangle has been rotated through 360 degrees.

```
10 R=40:S=25:G=π/18
20 PRINT"■":INPUT"INITIAL X,Y CO-ORDS ";A1,B1
30 INPUT"SECOND  X,Y CO-ORDS ";A2,B2
40 INPUT"THIRD   X,Y CO-ORDS ";A3,B3
50 INPUT"■TYPE ANGULAR STEP IN DEGREES ";AN
60 INPUT"■DO YOU WISH TO CLEAR THE TRACE EACH PLOT (Y OR N) ";V$
70 IF (V$="Y")+(V$="N") THEN 90
80 GOTO 60
90 AA=A1-R:BB=B1-S
100 GOSUB 260:R1=D:Q1=Q
110 AA=A2-R:BB=B2-S
120 GOSUB 260:R2=D:Q2=Q
130 AA=A3-R:BB=B3-S
140 GOSUB 260:R3=D:Q3=Q
150 PRINT"■"
160 FOR Q=0 TO 36+(V$="N")/10 STEP AN/10
170 X1=R1*COS(Q1+Q*G):Y1=R1*SIN(Q1+Q*G)
180 X2=R2*COS(Q2+Q*G):Y2=R2*SIN(Q2+Q*G)
190 X3=R3*COS(Q3+Q*G):Y3=R3*SIN(Q3+Q*G)
200 IF V$="Y" THEN PRINT"■"
210 M1=X1:M2=X2:N1=Y1:N2=Y2:GOSUB 290
220 M1=X2:M2=X3:N1=Y2:N2=Y3:GOSUB 290
230 M1=X3:M2=X1:N1=Y3:N2=Y1:GOSUB 290
240 FOR W=1 TO 100:NEXT W:NEXT Q
250 END
260 Q=π-π/2*SGN(BB)
270 D=SQR(AA*AA+BB*BB):IF AA<>0 THEN Q=ATN(BB/AA):IF AA<0 THEN Q=Q+π
280 RETURN
290 L=SQR((M1-M2)*(M1-M2)+(N1-N2)*(N1-N2))
300 FOR I=0 TO L+1
310 X=M1+I/(L+1)*(M2-M1):Y=N1+I/(L+1)*(N2-N1)
320 IF (R+X>=0)*(S+Y>=0)*(R+X<80)*(S+Y<50) THEN SET R+X,S+Y
330 NEXT
340 RETURN
```

Hoping these may be of interest.

D.B. LEE
ESSEX

****



Three Games Programs listings from **MR. FRANK WOODWARD.** These programs were written using one of the "BASIC MODIFIERS" and anyone keying these programs without a "TOOLKIT" will need to modify them accordingly. Some differences to notice are in lines where there are PRINT @ n,n and string equivalents etc.

```
1000 REM*GAME OF 'FRUSTRATION'*
1010 REM*WRITTEN FOR THE MZ-80K IN EXTENDED BASIC*
1020 REM*BY F.E.WOODWARD,8 VIOLET AV.RAMSGATE*
1030 REM*DEC.1981*
1040 REM*REQUIRES ABOUT 14K RAM TO RUN*
1050 REM*INITIALISATION OF VARIABLES*
1060 HS=0 : DIM GM(255,4)
1070 S$="×" : YM$="Y" : CM$="C"
1080 CS$="    " : CB$=SPC(16)
1090 YM=1:X1=0:Y1=0:M1=0:YS=0:X2=0:Y2=0:YZ=0
1100 CM=1:P1=0:Q1=0:M2=0:CS=0:P2=0:Q2=0:CZ=0
1110 T1$="KEY ANY NUMBER"
1120 T2$="FROM 1 TO 6 FOR"
1130 T3$="YOUR MOVE"
1140 T4$="MY MOVE IS"
1150 T5$="BEST SCORE SO FAR="
1160 PRINT"E"
1170 PRINT@2,11;"################"
1180 PRINT@3,11;"#              #"
1190 PRINT@4,11;"# FRUSTRATION #"
1200 PRINT@5,11;"#              #"
1210 PRINT@6,11;"################"
1220 GOSUB 3300
1230 PRINT@8,3;"THE AIM OF THE GAME IS TO MOVE FROM"
1240 PRINT@10,2;"THE TOP LEFT OF THE GRID TO THE BOTTOM"
1250 PRINT@12,2;"RIGHT.A MOVE CAN BE FROM 1 TO 6 CELLS."
1260 PRINT@14,3;"EACH CELL CAN CONTAIN A VALUE TO BE"
1270 PRINT@16,2;"TO THE PLAYERS SCORE,OR A TRAP"
1280 PRINT@18,2;"WHICH SENDS YOU BACK 3 OR 6 CELLS,
1290 PRINT@20,2;"OR IT MAY CONTAIN A BOMB,IN WHICH CASE"
1300 PRINT@22,2;"THE PLAYER IS FINISHED"
1310 GOTO1620
1320 PRINT"E"
1330 PRINT@2,3;"FOR EACH GAME YOU WILL BE PLAYING"
1340 PRINT@4,2;"AGAINST THE COMPUTER WHO CAN ONLY USE"
1350 PRINT@6,2;"RANDOM MOVES BETWEEN 1 AND 6"
1360 PRINT@8,3;"THE IDEA IS TO GET THE HIGHEST SCORE"
1370 PRINT@10,2;"YOU CAN.TO DO THIS TRY TO REMEMBER"
1380 PRINT@12,2;"WHERE HIGH SCORES ARE AND WHERE TRAPS"
1390 PRINT@14,2;"ARE AS IT MAY HELP YOU TO REVISIT SOME"
1400 PRINT@16,2;"CELLS.*NOTE:-EVERY GAME IS DIFFERENT &"
```

```
1410 PRINT@18,2;"CELLS CAN ONLY BE REVISITED A RANDOM"
1420 PRINT@20,2;"NO.OF TIMES & THEN THEY SCORE ZERO.*"
1430 PRINT@23,1;"DO YOU WISH TO GO FIRST (KEY Y or N)
1440 GET AN$
1450 IF AN$="" GOTO 1440
1460 PRINT"@"
1470 REM*SET UP MATRIX*
1480 FOR X=1 TO 15
1490 FOR Y=1 TO 15
1500 PRINT@X,Y;S$
1510 NEXT Y
1520 NEXT X
1530 PRINT@2,17;"THAT MOVE SCORED-"
1540 PRINT@4,17;"YOUR TOTAL IS"
1550 PRINT@6,17;"AFTER";@6,28;"MOVES"
1560 PRINT@7,17;"--------------------"
1570 PRINT@8,17;"MY MOVE SCORED---"
1580 PRINT@10,17;"MY TOTAL IS"
1590 PRINT@12,17;"AFTER";@12,28;"MOVES"
1600 IF AN$="Y" GOTO 1880
1610 IF AN$="N" GOTO 2200
1620 REM*SET GAME GRID AND CONTENTS*
1630 FOR X=1 TO 15
1640 FOR Y=1 TO 15
1650 N=(15*(X-1)+Y):GM(N,0)=X:GM(N,1)=Y
1660 NEXT Y
1670 NEXT X
1680 FOR I=1 TO 225
1690 GM(I,2)=INT(10*RND(1)+1)
1700 GM(I,3)=2+INT(3*RND(1)+1)
1710 GM(I,4)=0
1720 NEXT I
1730 FOR P=0 TO 210 STEP 15
1740 Z=P+INT(14*RND(1)+1)
1750 GM(Z,2)=30 :REM SET BOOBY TRAPS
1760 NEXT P
1770 FOR Q=0 TO 180 STEP 30
1780 Z=0+INT(30*RND(1)+1)
1790 IF GM(Z,2)>10 GOTO 1780
1800 GM(Z,2)=60 :REM SET BOOBY TRAPS
1810 NEXT Q
1820 R=INT(120*RND(1)+1)
1830 Z=60+R
1840 IF GM(Z,2)>10 GOTO 1820
1850 GM(Z,2)=80 :REM SET BOMB
1860 GOTO 1320
1870 REM*PLAYERS ROUTINE*
1880 FOR X=17 TO 23
1890 PRINT@X,1;CB$
1900 NEXT X
1910 PRINT@17,2;T1$;@19,2;T2$;@21,2;T3$
1920 GET YN
1930 IF (YN<1)+(YN>6) GOTO 1920
1940 X1=GM(YM,0):Y1=GM(YM,1)
1950 YM=YM+YN
1960 IF YM=>225 THEN PRINT@GM(225,0),GM(225,1);YM$:YF=1:GOTO 2630
1970 IF (X1=P1)*(Y1=Q1) THEN PRINT@X1,Y1;CM$ :GOTO 1990
1980 PRINT@X1,Y1;S$
1990 X2=GM(YM,0):Y2=GM(YM,1)
2000 IF  (X2=P1)*(Y2=Q1) THEN PRINT@X2,Y2;"B":GOTO 2020
2010 PRINT@X2,Y2;YM$
2020 GM(YM,4)=GM(YM,4)+1
2030 IF GM(YM,4)=GM(YM,3) THEN GM(YM,2)=0
2040 X1=X2:Y1=Y2:M1=M1+1
```

```
2050 PA=2000 : GOSUB 2990
2060 NS=GM(YM,2)
2070 IF NS=<10 THEN YS=YS+NS:GOTO 2130
2080 IF NS>60 THEN YZ=1:GOTO 2490
2090 GOSUB 3140
2100 YM=YM-NS/10
2110 IF YM<1 THEN YM=1
2120 GOTO 1980
2130 PRINT@2,34;CS$;@2,34;NS
2140 PRINT@4,32;CS$;@4,32;YS
2150 PRINT@6,23;CS$;@6,23;M1
2160 IF CZ=1 GOTO 1880
2170 IF CF=1 GOTO 1880
2180 GOTO 2200
2190 REM*COMPUTERS PLAY ROUTINE*
2200 FOR X=17 TO 23
2210 PRINT@X,1;CB$
2220 NEXT X
2230 CN=INT(6*RND(1)+1)
2240 PRINT@17,2;T4$;@17,14;CN
2250 P1=GM(CM,0):Q1=GM(CM,1)
2260 CM=CM+CN
2270 IF CM=>225 THEN PRINT@GM(225,0),GM(225,1);CM$:CF=1:GOTO 2630
2280 IF(P1=X1)*(Q1=Y1) THEN PRINT@P1,Q1;YM$:GOTO2300
2290 PRINT@P1,Q1;S$
2300 P2=GM(CM,0):Q2=GM(CM,1)
2310 IF(P2=X1)*(Q2=Y1) THEN PRINT@P2,Q2;"B":GOTO 2330
2320 PRINT@P2,Q2;CM$
2330 GM(CM,4)=GM(CM,4)+1
2340 IF GM(CM,4)=GM(CM,3) THEN GM(CM,2)=0
2350 P1=P2:Q1=Q2:M2=M2+1
2360 PA=2000 : GOSUB 2990
2370 NS=GM(CM,2)
2380 IF NS=<10 THEN CS=CS+NS:GOTO 2440
2390 IF NS>60 THEN CZ=1:GOTO 2490
2400 GOSUB 3140
2410 CM=CM-NS/10
2420 IF CM<1 THEN CM=1
2430 GOTO 2290
2440 PRINT@8,34;CS$;@8,34;NS
2450 PRINT@10,32;CS$;@10,32;CS
2460 PRINT@12,23;CS$;@12,23;M2
2470 IF YZ=1 GOTO 2200
2480 GOTO 1880
2490 REM*BOMBED OUT ROUTINE*
2500 GOSUB 2950
2510 GOSUB 3030
2520 GOSUB 2950
2530 IF (YZ=1)*(CZ=1) GOTO 2640
2540 IF YZ=1 GOTO 2590
2550 PRINT@14,18;"I'VE BOBBED OUT"
2560 PRINT@16,18;"NOW YOU PLAY ON TO"
2570 PRINT@18,18;"SEE WHAT YOU CAN SCORE"
2580 GOTO 1880
2590 PRINT@14,18;"YOU'VE BOMBED OUT"
2600 PRINT@16,18;"NOW I'LL PLAY ON TO"
2610 PRINT@18,18;"SEE WHAT I CAN SCORE"
2620 GOTO 2200
2630 REM*FINISHED ROUTINE*
2640 IF ((YF=1)+(YZ=1))*((CF=1)+(CZ=1)) THEN GOTO 2740
2650 IF YF=1 GOTO 2680
2660 IF CF=1 GOTO 2710
2670 GOSUB 2950
2680 PRINT@14,18;"YOUVE FINISHED"
2690 PRINT@16,18;"NOW WAIT FOR ME"
```

```
2700 GOTO 2200
2710 PRINT@14,18:"I'VE FINISHED"
2720 PRINT@16,18:"I WILL WAIT FOR YOU"
2730 GOTO 1880
2740 GOSUB 2950
2750 PRINT@14,18:"WE HAVE BOTH FINISHED"
2760 PA=2000:GOSUB 3000
2770 GOTO 2790
2780 REM*FINAL SCORE & RESULT*
2790 GOSUB 2950
2800 GOSUB 3140
2810 IF VS>CS GOTO 2840
2820 PRINT@14,18:"SORRY! I WON":IF CS>HS THEN HS=CS
2830 GOTO 2850
2840 PRINT@14,18:"WELL DONE! YOU WON : IF VS>HS THEN HS=VS
2850 PRINT@16,18:"HIGHEST SCORE SO FAR=";@17,30;HS
2860 PRINT@19,18:"DO YOU WANT ANOTHER"
2870 PRINT@21,18:"GAME(KEY Y OR N)
2880 GET RP$
2890 IF RP$="Y" GOTO 1060
2900 IF RP$="N" GOTO 2920
2910 GOTO 2880
2920 PRINT"@":PRINT@5,3:"SO-LONG FOR NOW THEN"
2930 END
2940 REM*CLEAR BLOCK ROUTINE*
2950 FOR C=13 TO 22
2960 PRINT@C,16;SPC(22)
2970 NEXT C
2980 RETURN
2990 REM*PAUSE ROUTINE*
3000 FOR P=1 TO PA : NEXT
3010 RETURN
3020 REM*BOMB BURST ROUTINE*
3030 B1$=" ":B2$="*"
3040 FOR BN=1 TO 4
3050 U=15:V=25
3060 FOR N=0 TO 4
3070 PRINT@U-N,V-N;B2$;@U-N,V+N;B2$;@U+N,V-N;B2$;@U+N,V+N;B2$
3080 FOR A=1 TO 5:POKE 4514,A:USR(68):NEXT A:USR(71)
3090 PRINT@U-N,V-N;B1$;@U-N,V+N;B1$;@U+N,V-N;B1$;@U+N,V+N;B1$
3100 FOR A=5 TO 1 STEP -1:POKE 4514,A:USR(68):NEXT A:USR(71)
3110 NEXT N
3120 NEXT BN
3130 RETURN
3140 REM*BOOBY TRAP GRAPHICS & SOUND*
3150 FOR N=1 TO 5
3160 PRINT@15,20:"BOOBY TRAP"
3170 FOR A=15 TO 10 STEP -1:POKE 4514,A:USR(68):NEXT A:USR(71)
3180 PRINT@15,20:"          "
3190 FOR A=10 TO 15:POKE 4514,A:USR(68):NEXT A:USR(71)
3200 NEXT N
3210 GOSUB 2950
3220 RETURN
3230 REM*GAME WON SOUND EFFECTS*
3240 FOR A=10 TO 1 STEP -1
3250 POKE 4514,A
3260 FOR B=0 TO 255 STEP A
3270 POKE 4513,B:USR(68)
3280 NEXT B,A:USR(71)
3290 RETURN
3300 REM*START SOUNDS*
3310 FOR T= 1 TO 8
3320 FOR A=-255 TO 255 STEP 30:B=ABS(A):POKE 4514,1:POKE 4513,B
3330 USR(68):NEXT A:MUSIC"R2"
3340 NEXT T
3350 RETURN
```

```
1000 REM*WITCHES FORTRESS ADVENTURE PROGRAM
1010 REM*WRITTEN IN SHARP EXTENDED BASIC
1020 REM*BY F.E.WOODWARD,8 VIOLET AV.RAMSGATE
1030 PRINT"E"
1040 CLR:RESTORE:DIM DR(19),PG(5),EV(12),SU(16,19),R(12,1),A$(5),MO(250,1)
1050 VT=(50*INT(3*RND(1)+1))+50
1060 FOR N=1 TO 5 : PG(N)=0 : NEXT N
1070 M1$="KEY YOUR MOVE IN"
1080 M2$="NOW(As in Key dias)"
1090 M3$="YOU CANT GO THROUGH"
1100 M4$="THE WALL.TRY AGAIN"
1110 M5$="YOU HAVE LANDED ON"
1120 M6$="YOU'VE ENCOUNTERED"
1130 M7$="YOU'VE AWAKENED"
1140 M8$="YOU'VE RELEASED"
1150 M9$="YOU HAVE FOUND"
1160 H2$="A PROTECTED D00R"
1170 H1$="YOUR REPELLED"
1180 H3$="A FORCE FIELD"
1190 H4$="A GIANT SERPENT"
1200 H5$="THE DRAGON"
1210 H6$="THE EVIL WITCH"
1220 H7$="A FLOOR TRAP"
1230 H8$="A FORCE BEAM"
1240 H9$="A POISON DART"
1250 A$(1)="A MAGIC SHIELD"
1260 A$(2)="A SWORD"
1270 A$(3)="AN ELIXIR"
1280 A$(4)="A SUIT OF ARMOUR"
1290 A$(5)="A MAGIC CHARM"
1300 PRINT@ 3,9;"   ▲          ▲"
1310 PRINT@ 4,9;"  ▲▲▲        ▲▲▲"
1320 PRINT@ 5,9;" |   |      |   |"
1330 PRINT@ 6,9;" | ▒ |      | ▒ |"
1340 PRINT@ 7,9;" |   ▒▒▒▒▒▒▒▒   |"
1350 PRINT@ 8,9;" | ▒ |      | ▒ |"
1360 PRINT@ 9,9;" |   | ▒ ▒ ▒ |   |"
1370 PRINT@10,9;" | ▒ | ▒ ▒ ▒ | ▒ |"
1380 PRINT@11,9;" | ▒ |      |   |"
1390 PRINT@12,9;"  ▔▔▔▔▔▔▔▔▔▔▔▔▔▔"
1400 FOR T=1 TO 15:USR(62):NEXT T
1410 PRINT@15,7;"THE WITCHES FORTRESS"
1420 PRINT@16,7;"▔▔▔▔▔▔▔▔▔▔▔▔▔▔▔▔▔▔▔▔"
1430 GOSUB 5570:PRINT"E"
1440 PRINT@4,2;"You are at the entrance of the WITCHES"
1450 PRINT@5,1;"FORTRESS.There are 12 rooms each of"
1460 PRINT@6,1;"which has 9 cells in it,and you must"
1470 PRINT@7,1;"search each cell for the TREASURE."
1480 PRINT@9,2;"Some of the cells contain hazards,and"
1490 PRINT@10,1;"some gains.Also some of the doors are"
1500 PRINT@11,1;"repellers."
1510 PRINT@13,2;"You have 200 moves to find the"
1520 PRINT@14,1;"treasure,and you must find it befor"
1530 PRINT@15,1;"Your vitality runs out."
1540 PRINT@18,1;"KEY'S' WHEN YOU ARE READY TO START"
1550 PRINT@20,1;"AND WAIT A FEW SECONDS"
1560 GET SS$:IF SS$<>"S" GOTO 1560
1570 PRINT"E"
1580 PRINT@18,1;"T Y U KEY AS SHOWN"
1590 PRINT@19,1;" \|/"
1600 PRINT@20,1;"G-H-J TO MOVE IN ANY"
1610 PRINT@21,1;" /|\"
1620 PRINT@22,1;"B N M DIRECTION"
1630 GOTO5760
1640 REM*DOOR TRAP ROUTINE*
```

```
1650 GOSUB 5600
1660 PRINT@1,21;"
1670 PRINT@2,21;"
1680 PRINT@3,21;"
1690 PRINT@4,21;"
1700 PRINT@5,21;"
1710 PRINT@6,21;"
1720 PRINT@7,21;"
1730 PRINT@8,21;"
1740 PRINT@9,21;"
1750 PRINT@10,21;"
1760 PRINT@11,21;"
1770 PRINT@12,21;"
1780 PRINT@13,21;"
1790 PRINT@14,21;"
1800 PRINT@15,21;"
1810 PRINT@16,21;"
1820 FOR T=1 TO 10:FOR A=20 TO 25:POKE 4514,A:USR(68):NEXT A:USR(71):NEXT T
1830 PRINT@18,21;M6$;@19,21;H2$;@21,21;H1$;@22,21;"TWO STEPS"
1840 GOSUB 5570: BT=ST-2 : IF BT=<0 THEN BT=1
1850 X=MO(BT,0) : Y=MO(BT,1) : PRINT@P,Q;" " : PRINT@X,Y;P$
1860 IF PG(4)>0 GOTO 1900
1870 MO(ST,0)=X : MO(ST,1)=Y : N=SU(X,Y) : IF N>1 GOTO 6530
1880 GOSUB 5600
1890 RETURN
1900 GOSUB 5600
1910 PRINT@2,21;"HOWEVER SINCE YOU"
1920 PRINT@4,21;"WERE WEARING YOUR"
1930 PRINT@6,21;"ARMOUR YOU OVER-"
1940 PRINT@8,21;"LOADED THE DOOR"
1950 PRINT@10,21;"MECHANISM.THE WAY"
1960 PRINT@11,21;"IS NOW OPEN"
1970 SU(P,Q)=0 : GOSUB 5570: GOTO 1870
1980 REM*CORRIDOR ROUTINE*
1990 GOSUB 5600
2000 PRINT@ 1,21;"
2010 PRINT@ 2,21;"
2020 FOR K=3 TO 10 : PRINT@K,21;"
2030 PRINT@11,21;"
2040 PRINT@12,21;"
2050 PRINT@14,21;M6$;@16,21;H3$
2060 PRINT@18,21;"YOU CANT PASS BUT";@20,21;"MUST FIND ANOTHER"
2070 PRINT@22,21;"WAY ROUND"
2080 FOR KS=1 TO 3
2090 FOR L=3 TO 10
2100 PRINT@L,24;":GOSUB 2170
2110 PRINT@L,24;"
2120 PRINT@L,24;":GOSUB 2170
2130 PRINT@L,24;"
2140 NEXT L
2150 NEXT KS
2160 GOTO 2180
2170 FOR A=1 TO 5:POKE 4514,A:USR(68):NEXT A:USR(71):RETURN
2180 BT=ST-1:X=MO(BT,0):Y=MO(BT,1)
2190 PRINT@P,Q;" ";@X,Y;P$
2200 MO(ST,0)=X:MO(ST,1)=Y
2210 RETURN
2220 REM*SERPENT ROUTINE*
2230 GOSUB 5600
2240 PRINT@ 14,21;M8$;@16,21;H4$
2250 PRINT@3,24;";@4,24;";@5,34;";@6,35;";@7,34;"
2260 PRINT@8,22;";@9,22;";@10,35;";@11,34;"
2270 PRINT@12,22;";@10,22;";@11,22;";@5,25;"
2280 FOR W=1 TO 8:PRINT@4,21;";@4,21;"  ";@4,21;" : NEXT W
```

```
2290 FOR T=1 TO 4:FOR A=5 TO 25:POKE 4514,A:USR(68):NEXT A:USR(71):NEXT T
2300 GOSUB 5570: GOSUB 5600
2310 IF (PG(2)>0)*(PG(4)>0) GOTO 2410
2320 IF PG(1)>0 GOTO 2460
2330 IF PG(4)>0 GOTO 2530
2340 PRINT@2,21;"YOU ARE INJURED &"
2350 PRINT@4,21;"MUST BACK OFF"
2360 BT=ST-3 : X=MO(BT,0) : Y=MO(BT,1)
2370 PRINT@P,Q;" ";@X,Y;P$
2380 MO(ST,0)=X : MO(ST,1)=Y : VT=VT-30
2390 N=SU(X,Y) : IF N>1 GOTO 6530
2400 GOTO 2540
2410 PRINT@2,21;"WITH YOUR ARMOUR"
2420 PRINT@4,21;"AND SWORD,YOU"
2430 PRINT@6,21;"OVERCAME THE GIANT"
2440 PRINT@8,21;"SERPENT"
2450 SU(P,Q)=0 : MO(ST,0)=P : MO(ST,1)=Q : X=P : Y=Q : GOTO 2540
2460 PRINT@2,21;"YOUR MAGIC SHIELD"
2470 PRINT@4,21;"ALLOWED YOU TO GET"
2480 PRINT@6,21;"PAST THE SERPENT"
2490 PRINT@8,21;"BUT IT IS UNHARMED"
2500 PRINT@10,21;"AND YOU RECIEVED"
2510 PRINT@11,21;"SLIGHT INJURIES"
2520 VT=VT-10 : MO(ST,0)=P : MO(ST,1)=Q : X=P : Y=Q : GOTO 2540
2530 PRINT@2,21;"YOUR ARMOUR HAS" : GOTO 2470
2540 GOSUB 5570: GOSUB 5600: RETURN
2550 REM*DRAGON ROUTINE*
2560 GOSUB 5600
2570 PRINT@ 2,20;"      "
2580 PRINT@ 3,20;"      "
2590 PRINT@ 4,20;"      "
2600 PRINT@ 5,20;"      "
2610 PRINT@ 6,20;"      "
2620 PRINT@ 7,20;"      "
2630 PRINT@ 8,20;"      "
2640 PRINT@ 9,20;"      "
2650 PRINT@10,20;"      "
2660 PRINT@11,20;"      "
2670 PRINT@12,20;"      "
2680 PRINT@13,20;"      "
2690 PRINT@14,20;"      "
2700 PRINT@15,20;"      "
2710 PRINT@16,20;"      "
2720 PRINT@17,20;"      "
2730 PRINT@18,20;"      "
2740 FOR T=1 TO 2:FOR A=50 TO 200:POKE 4514,A:USR(68):NEXT A:USR(71):NEXT T
2750 PRINT@20,21;M7$;@22,21;H5$
2760 GOSUB 5570: GOSUB 5600
2770 IF PG(5)>0 AND PG(2)>0 GOTO 2860
2780 IF PG(5)>0 GOTO 2890
2790 IF PG(2)>0 GOTO 2930
2800 IF PG(1)>0 GOTO 2970
2810 PRINT@4,21;"YOU MANAGED TO";@6,21;"ESCAPE BUT WERE "
2820 PRINT@8,21;"WOUNDED AND HAD TO"
2830 PRINT@10,21;"RETREAT"
2840 VT=VT-40 : X=MO(ST-4,0) : Y=MO(ST-4,1)
2850 PRINT@P,Q;" ";@X,Y;P$ : MO(ST,0)=X: MO(ST,1)=Y : GOTO 3040
2860 PRINT@2,21;"WITH YOUR SWORD AND"
2870 PRINT@4,21;"ARMOUR YOU OVERCAME "
2880 PRINT@6,21;"THE DRAGON ":SU(P,Q)=0 : GOTO 3030
2890 PRINT@2,21;"YOUR ARMOUR SAVED"
2900 PRINT@4,21;"YOU,AND YOU ESCAPED"
2910 PRINT@6,21;"BUT LEFT THE DRAGON"
2920 PRINT@8,21;"UNHARMED" : GOTO 3030
2930 PRINT@2,21;"WITH YOUR SWORD YOU"
```

```
2940 PRINT@4,21;"HELD OFF THE DRAGON"
2950 PRINT@6,21;"BUT WERE INJURED"
2960 VT=VT-20 : GOTO 3030
2970 PRINT@2,21;"YOUR MAGIC SHIELD"
2980 PRINT@4,21;"KEPT THE DRAGON AT"
2990 PRINT@6,21;"BAY,BUT YOU HAD TO"
3000 PRINT@8,21;"RETREAT"
3010 X=MO(ST-3,0) : Y=MO(ST-3,1) : PRINT@P,Q;" ";@X,Y;P$
3020 MO(ST,0)=X : MO(ST,1)=Y : GOTO 3040
3030 MO(ST,0)=X : MO(ST,1)=Y : X=P : Y=Q: GOTO 3050
3040 IF SU(X,Y)>1 THEN N=SU(X,Y) : GOTO 6530
3050 GOSUB 5570: GOSUB 5600: RETURN
3060 REM*WITCH ROUTINE*
3070 GOSUB 5600
3080 PRINT@ 2,23;"      ◢"
3090 PRINT@ 3,23;"     ◢▀"
3100 PRINT@ 4,23;"    ◢▀▀"
3110 PRINT@ 5,23;"   ◢▀▀▀"
3120 PRINT@ 6,23;"  / @@@"
3130 PRINT@ 7,23;" /  @@@"
3140 PRINT@ 8,23;"/  /  @@"
3150 PRINT@ 9,23;"┐ /   @@@"
3160 PRINT@10,23;" ┌  ┌ / @@"
3170 PRINT@11,23;" └┌─  |"
3180 PRINT@12,23;"  / ~~\"
3190 PRINT@13,23;"  | *\  \\"
3200 PRINT@14,23;"  ▌* / /  \"
3210 PRINT@15,23;"  |▙▟/    \"
3220 PRINT@16,23;"  ▌ * * * |"
3230 FOR T=1 TO 3
3240 FOR A=-255 TO 255 STEP 8:B=ABS(A):POKE 4514,B:POKE 4513,B:USR(68)
3250 NEXT A:USR(71)
3260 NEXT T
3270 PRINT@20,21;M6$;@22,21;H6$
3280 GOSUB 5570:GOSUB 5600
3290 IF PG(5)>0 GOTO 3400
3300 PRINT@2,21;"THE WITCH CAST A"
3310 PRINT@4,21;"SPELL OVER YOU"
3320 PRINT@6,21;"YOU WAKE AFTER A"
3330 PRINT@8,21;"TEN MOVE PERIOD IN"
3340 PRINT@10,21;"THE CORRIDOR AND"
3350 PRINT@12,21;"YOU HAVE LOST SOME"
3360 PRINT@13,21;"VITALITY"
3370 MO(ST,0)=7:MO(ST,1)=10:X=7:Y=10
3380 FOR ST=ST TO ST+10:MO(ST,0)=7:MO(ST,1)=10:NEXT
3390 PRINT@P,Q;" ";@X,Y;P$:VT=VT-30:P=X:Q=Y:GOTO 3460
3400 PRINT@2,21;"YOUR MAGIC CHARM"
3410 PRINT@4,21;"PROTECTS YOU FROM"
3420 PRINT@6,21;"HER EVIL SPELL &"
3430 PRINT@8,21;"YOU MAY PASS ON"
3440 PRINT@10,21;"YOUR WAY"
3450 MO(ST,0)=P:MO(ST,1)=Q:X=P:Y=Q
3460 GOSUB 5570:GOSUB 5600:RETURN
3470 REM*FLOOR TRAP ROUTINE*
3480 GOSUB 5600:PRINT@17,21;M5$;@19,21;H7$
3490 PRINT@ 2,22;"◣──────◢"
3500 PRINT@ 3,22;"▙▖    ▗▟"
3510 PRINT@ 4,22;"▙▙▖  ▗▟▟"
3520 PRINT@ 5,22;"▙▙▙  ▟▟▟"
3530 PRINT@ 6,22;"▙▙▙  ▟▟▟"
3540 PRINT@ 7,22;"▙▙▙  ▟▟▟"
3550 PRINT@8,22;"▙▙▙  ▟▟▟"
3560 PRINT@9,22;"▙▙▙  ▟▟▟"
3570 PRINT@10,22;"▙▙▙  ▟▟▟"
3580 PRINT@11,22;"▙▙▙──▟▟▟"
```

```
4890 PRINT@ 5,24;"  ▲"
4900 PRINT@ 6,24;"▲■E■▲"
4910 PRINT@ 7,24;"■■L■■"
4920 PRINT@ 8,24;"■■I■■"
4930 PRINT@ 9,24;"■■X■■"
4940 PRINT@10,24;"■■I■■"
4950 PRINT@11,24;"■■R■■"
4960 PRINT@12,24;"■■■■■"
4970 PRINT@13,24;"▼■■■▼"
4980 FOR A=255 TO 0 STEP-12:POKE4514,1:POKE 4513,A:USR(68):NEXT A:USR(71)
4990 FOR A=-255 TO 255 STEP 8:B=ABS(A):POKE 4514,1:POKE 4513,B:USR(68)
5000 NEXT A:USR(71)
5010 PRINT@15,21;M9$;@17,21;A$(3)
5020 PRINT@17,21;"YOU DRAIN IT ALL"
5030 PRINT@19,21;"& GO ON WITH NEW"
5040 PRINT@21,21;"VITALITY"
5050 MO(ST,0)=P:MO(ST,1)=Q:X=P:Y=Q
5060 GOSUB 5570:GOSUB5600
5070 PRINT@2,21;"YOU MAY RETURN FOR"
5080 PRINT@3,23;"ANOTHER DRINK"
5090 PRINT@4,21;"WHENEVER YOU WISH"
5100 GOSUB 5570:GOSUB 5600:RETURN
5110 REM*ARMOUR ROUTINE*·
5120 GOSUB 5600
5130 PRINT@ 2,23;"    ▲■▲"
5140 PRINT@ 3,23;"    ■■■"
5150 PRINT@ 4,23;"    ■■■"
5160 PRINT@ 5,23;"    ▼■▼"
5170 PRINT@ 6,23;"  ▲■■■■■▲"
5180 PRINT@ 7,23;" ▲■■■■■■■▲"
5190 PRINT@ 8,23;"■ ▼■■■▼ ■"
5200 PRINT@ 9,23;"■  ■■■  ■"
5210 PRINT@10,23;"   ▲■■■▲"
5220 PRINT@11,23;"   ■■■■■"
5230 PRINT@12,23;"    ■ ■"
5240 PRINT@13,23;"    ■ ■"
5250 PRINT@14,23;"    ■ ■"
5260 PRINT@15,23;"    ■ ■"
5270 PRINT@16,23;"  ▲▼ ▼▲"
5280 FOR A=20 TO 10 STEP-1:POKE 4514,A:FOR B=100 TO 255 STEP A:POKE 4513,B
5290 USR(68):NEXT B,A:USR(71)
5300 PRINT@18,21;M9$;@20,21;A$(4):PG(4)=1
5310 GOSUB 5570:GOSUB 5600
5320 PRINT@2,21;"YOU PUT IT ON &"
5330 PRINT@4,21;"GO ON YOUR WAY WITH"
5340 PRINT@6,21;"A NEW IMMUNITY"
5350 MO(ST,0)=P:MO(ST,1)=Q:X=P:Y=Q
5360 GOSUB 5570:GOSUB 5600:RETURN
5370 REM*MAGIC CHARM ROUTINE*
5380 GOSUB 5600
5390 PRINT@ 2,22;"      V "
5400 PRINT@ 3,22;"      A "
5410 PRINT@ 4,22;"      V "
5420 PRINT@ 5,22;"     ▲■▲"
5430 PRINT@ 6,22;"    ▲■■■▲"
5440 PRINT@ 7,22;"    ■■■■■"
5450 PRINT@ 8,22;"  ▲■■■■■■■▲"
5460 PRINT@ 9,22;" ▲■■■■■■■■■▲"
5470 PRINT@10,22;"▼■■■■■■■■■▼"
5480 PRINT@11,22;" ▼■■■■■■■▼"
5490 PRINT@12,22;"    ■■■■■"
5500 PRINT@13,22;"    ▼■■■▼"
5510 PRINT@14,22;"     ▼■▼"
5520 FOR T=1 TO 10:FOR A=10 TO 5 STEP-1:POKE 4514,A:USR(68):NEXT A:USR(71)
5530 NEXT T
```

```
4240 PRINT@4,21;"DRANK NEUTRALISED"
4250 PRINT@6,21;"THE POISON":PG(3)=0
4260 GOTO 4310
4270 PRINT@2,21;"YOUR ARMOUR SAVED"
4280 PRINT@4,21;"YOU FROM MOST OF"
4290 PRINT@6,21;"THE POISON"
4300 VT=VT-5
4310 MO(ST,0)=P:MO(ST,1)=Q:X=P:Y=Q
4320 GOSUB 5570:GOSUB 5600:RETURN
4330 REM*TREASURE*
4340 GOSUB 5600
4350 PRINT@ 3,21;"  \   \  /    /"
4360 PRINT@ 4,21;"   \       /"
4370 PRINT@ 5,21;"\ \  ▄▄▄▄▄  / /"
4380 PRINT@ 6,21;"  ▄▄▄▄▄▄▄▄▄▄▄"
4390 PRINT@ 7,21;"\ ▄▄▄▄▄▄▄▄▄▄▄  /"
4400 PRINT@ 8,21;"  ▄▄▄▄▄▄▄▄▄▄▄"
4410 PRINT@ 9,21;"\ ▄▄▄▄▄▄▄▄▄▄▄  /"
4420 PRINT@10,21;"  ▄▄ TREASURE ▄▄"
4430 PRINT@11,21;"/ ▄▄▄▄▄▄▄▄▄▄▄  \"
4440 PRINT@12,21;"  ▄▄▄▄▄▄▄▄▄▄▄"
4450 PRINT@13,21;"/             \"
4460 PRINT@14,21;"  /  / / \  \  \"
4470 FOR T=1 TO 2:FOR A=10 TO 1 STEP-1:POKE 4514,A:FOR B=0 TO 255 STEP A
4480 POKE 4513,B:USR(68):NEXTB,A:USR(71):NEXT T
4490 PRINT@16,21;M9$;@18,21;"THE TREASURE"
4500 GOSUB 5570:GOSUB 5600
4510 PRINT@4,21;"ALL HAZARDS ARE"
4520 PRINT@6,21;"NOW NEUTRALISED"
4530 PRINT@8,21;"THE WAY OUT IS"
4540 PRINT@10,21;" NOW CLEAR"
4550 GOSUB 5570:GOSUB 5600:GOTO 6200
4560 REM*SHIELD ROUTINE*
4570 GOSUB 5600
4580 PRINT@4,25;"┌\   /┐"
4590 PRINT@5,25;"| ● \ * |"
4600 PRINT@6,25;"| ●●  ** |"
4610 PRINT@7,25;"| ●●●*** |"
4620 PRINT@8,25;"\     /"
4630 PRINT@9,25;" \ ▯▯ /"
4640 PRINT@10,25;"  \  /"
4650 PRINT@11,25;"   \/"
4660 FOR A=5 TO 1 STEP-1:POKE 4514,A:FOR B=0 TO 150 STEP A:POKE 4513,B:US
4670 NEXT B,A:USR(71)
4680 PRINT@14,21;M9$;@16,21;A$(1):PG(1)=1:GOTO 4790
4690 REM*SWORD ROUTINE*
4700 GOSUB 5600
4710 PRINT@2,26;"▄▄"
4720 PRINT@3,26;" ▄▄ "
4730 PRINT@4,26;" ▄▄ "
4740 PRINT@5,26;"▄▄▄"
4750 FOR SS=6 TO 18 :PRINT@SS,26;" ▄▄ ":NEXT: PRINT@19,26;" ▄▄"
4760 FOR A=1 TO 10:POKE 4514,A:FOR B=0 TO 175 STEP A:POKE 4513,B:USR(68)
4770 NEXT B,A:USR(71)
4780 PRINT@20,21;M9$;@22,21;A$(2):PG(2)=1
4790 GOSUB 5570:GOSUB 5600
4800 PRINT@2,21;"YOU CLASP IT IN"
4810 PRINT@4,21;"YOUR HAND AND GO ON"
4820 PRINT@6,21;"WITH NEW HOPE"
4830 X=P:Y=Q:MO(ST,0)=X:MO(ST,1)=Y
4840 GOSUB 5570:GOSUB 5600:RETURN
4850 REM*ELIXIR ROUTINE*
4860 GOSUB 5600:VT=VT+20:PG(3)=1
4870 PRINT@ 3,24;"  ▄▄"
4880 PRINT@ 4,24;"  ▄▄"
```

```
3590 PRINT@12,22;"▛▀▀▀▀▀▀▀▀▀▀▀▀▀▀▜"
3600 PRINT@13,22;"▙▄▄▄▄▄▄▄▄▄▄▄▄▄▄▟"
3610 FOR A=100 TO 250:POKE 4514,A:USR(68):NEXT A:USR(71)
3620 FOR A=250 TO 100 STEP -1 :POKE 4514,A:USR(68):NEXT A:USR(71)
3630 GOSUB 5570:GOSUB 5600
3640 PRINT@2,21;"YOU HAVE FALLEN"
3650 PRINT@4,21;"INTO A TUNNEL &"
3660 PRINT@6,21;"THE ONLY WAY OUT"
3670 PRINT@8,21;"IS AT THE ENTRANCE"
3680 PRINT@10,21;"YOU STARTED FROM"
3690 IF PG(4)>0 GOTO 3730
3700 PRINT@12,21;"& YOU LOSE VITALITY"
3710 PRINT@14,21;"IN GETTING OUT"
3720 VT=VT-20:GOTO 3760
3730 PRINT@12,21;"BUT YOUR ARMOUR"
3740 PRINT@14,21;"SAVES YOU FROM"
3750 PRINT@16,21;"INJURY"
3760 PRINT@P,Q;" ";@16,10;P$
3770 MO(ST,0)=16:MO(ST,1)=10:X=16:Y=10
3780 GOSUB 5570:GOSUB 5600:RETURN
3790 REM*FORCE BEAM ROUTINE*
3800 GOSUB 5600
3810 PRINT@ 1,22;"▛▀▀▀▀▀▀▀▀▀▀▀▀▀▜| "
3820 PRINT@ 2,22;"||▛▀▀▀▀▀▀▀▀▀▀▀▜|| "
3830 PRINT@ 3,22;"|||▛▀▀▀▀▀▀▀▀▀▜||| "
3840 FOR LL=4 TO 12:PRINT@LL,22;"||||          ||||":NEXT
3850 PRINT@13,22;"|||▙▄▄▄▄▄▄▄▄▄▟||| "
3860 PRINT@14,22;"||▙▄▄▄▄▄▄▄▄▄▄▄▟|| "
3870 PRINT@15,22;"▙▄▄▄▄▄▄▄▄▄▄▄▄▄▟| "
3880 PRINT@18,21;M6$;@20,21;H3$
3890 FOR ZX=1 TO 5:FOR A=3 TO 8:POKE 4514,A:USR(68):NEXT A:USR(71)
3900 FOR FC=27 TO 30:FOR FB=4 TO 12:PRINT@FB,FC;"▒▒▒";@FB,FC;"   "
3910 NEXT FB:NEXT FC:NEXT ZX
3920 GOSUB 5600
3930 IF PG(4)>0 GOTO 4020
3940 PRINT@2,21;"YOU WERE STUNNED"
3950 PRINT@4,21;"BY THE FORCE BEAM"
3960 PRINT@6,21;"AND THROWN BACK"
3970 PRINT@8,21;"THREE MOVES"
3980 X=MO(ST-3,0):Y=MO(ST-3,1):MO(ST,0)=X:MO(ST,1)=Y:PRINT@P,Q;" ";@X,Y;P$
3990 VT=VT-15
4000 IF SU(X,Y)>1 THEN N=SU(X,Y):GOTO 6530
4010 GOTO 4070
4020 PRINT@2,21;"YOUR ARMOUR REDUCED"
4030 PRINT@4,21;"THE EFFECT OF THE"
4040 PRINT@6,21;"BEAM A & YOU ARE"
4050 PRINT@8,21;"ABLE TO GO ON"
4060 VT=VT-5:MO(ST,0)=P:MO(ST,1)=Q:X=P:Y=Q
4070 GOSUB 5570:GOSUB 5600:RETURN
4080 REM*POISON DART ROUTINE*
4090 GOSUB 5600
4100 PRINT@8,21;"—▬▬▬▬▬▬▬▬▬▬▬▬▬▬"
4110 PRINT@9,21;"                ▀▀▀▄"
4120 PRINT@7,21;"                ▄▄▄▀"
4130 POKE 4514,2:FOR B=0 TO 254 STEP 2:USR(68):NEXT B:USR(71)
4140 PRINT@14,21;M8$:@16,21;H9$
4150 GOSUB 5570:GOSUB 5600
4160 IF PG(4)>0 GOTO 4270
4170 IF PG(3)>0 GOTO 4230
4180 PRINT@2,21;"YOU ARE HIT BY"
4190 PRINT@4,21;"THE DART AND ARE "
4200 PRINT@6,21;"WEAKENED BY THE"
4210 PRINT@8,21;"POISON"
4220 VT=VT-20:GOTO 4310
4230 PRINT@2,21;"THE ELIXIR YOU"
```

```
5540 PRINT@16,22;M9$;@18,22;A$(5):PG(5)=1
5550 MO(ST,0)=P:MO(ST,1)=Q:X=P:Y=Q
5560 GOSUB 5570:GOSUB 5600:RETURN
5570 REM*PAUSE ROUTINE*
5580 FOR PA=1 TO 2000 : NEXT PA
5590 RETURN
5600 REM*CLEAR RT/HAND SIDOF SCREEN*
5610 FOR C=1 TO 22
5620 PRINT@C,21;SPC(19)
5630 NEXT C
5640 RETURN
5650 REM*STATUS DISPLAY*
5660 PRINT@1,25;"YOUR STATUS"
5670 PRINT@3,21;"YOUR VITALITY IS ->"
5680 PRINT@5,25;VT;@5,30;"POINTS"
5690 PRINT@7,21;"YOUR ATTRIBUTES ->"
5700 FOR Z=1 TO 5
5710 IF PG(Z)>0 THEN PRINT@8+Z,21;A$(Z)
5720 NEXT Z
5730 PRINT@15,21;"YOU HAVE HAD"
5740 PRINT@17,25;ST;@17,29;"MOVES"
5750 RETURN
5760 REM*SET DOOR TRAPS IN DR(19)
5770 FOR X=1 TO 19:DR(X)=0:NEXT X
5780 FOR Z=1 TO 6
5790 N=INT(19*RND(1)+1)
5800 IF DR(N)<>0 GOTO 5790
5810 DR(N)=2
5820 NEXT Z
5830 REM SET UP INITIAL SU MATRIX AND DISPLAY PLAN
5840 N=0
5850 FOR X=1 TO 16
5860 FOR Y=1 TO 19
5870 READ SU(X,Y)
5880 IF SU(X,Y)=1 THEN PRINT@X,Y;"■"
5890 IF SU(X,Y)=2 THEN N=N+1:SU(X,Y)=DR(N)
5900 NEXT Y
5910 NEXT X
5920 REM*SET CORRIDOR BARRIER
5930 B=INT(2*RND(1)+1)
5940 IF B=1 THEN SU(6,6)=3
5950 IF B=2 THEN SU(6,14)=3
5960 REM*SET EVENT MATRIX(EV(12))*
5970 FOR N=1 TO 12:EV(N)=0:NEXT N
5980 FOR X=4 TO 15
5990 N=INT(12*RND(1)+1)
6000 IF EV(N)<>0 GOTO 5990
6010 EV(N)=X
6020 NEXT X
6030 REM*SET UP ROOM CONTENTS CODE ie.POSSITION OF EV(N)*
6040 FOR N=1 TO 12
6050 READ R(N,0):READ R(N,1)
6060 NEXT N
6070 FOR J=1 TO 12
6080 X=R(J,0):Y=R(J,1)
6090 A=INT(2*RND(1)): B=INT(2*RND(1))
6100 X=X+A : Y=Y+B
6110 SU(X,Y)=EV(J)
6120 NEXT J
6130 GOTO 6310
6140 REM*LOST GAME*
6150 PRINT"■":PRINT@2,14;"HARD LUCK"
6160 FOR A=-150 TO 150:B=ABS(A):POKE 4514,B:USR(68):NEXT A:USR(71)
6170 IF ST=<0 GOTO 6190
6180 PRINT@10,5;"YOU DIED AFTER";@10,25;ST;@10,30;"MOVES":GOTO 6240
```

```
6190 PRINT@10,5;"YOU HAVE RUN OUT OF MOVES":GOTO 6240
6200 REM*WON GAME*
6210 PRINT"E":PRINT@2,4;"YOU REACHED THE END OF YOUR QUEST IN"
6220 PRINT@6,4;ST;@6,10;"MOVES,AND WITH"
6230 PRINT@10,4;VT;@10,10;"VITALITY POINTS LEFT"
6240 PRINT@18,2;"DO YOU WANT ANOTHER GAME (Y/N)"
6250 GET AW$
6260 IF AW$="Y" THEN PRINT"E":GOTO 1040
6270 IF AW$="N" GOTO 6290
6280 GOTO 6250
6290 PRINT"E":PRINT@10,5;"SO-LONG FOR NOW THEN"
6300 END
6310 REM*MOVE ROUTINE AND GAME START*
6320 P$=CHR$(99) :X=16 : Y=10 : PRINT@X,Y;P$
6330 GOSUB 5600: GOSUB 5650: GOSUB 5570
6340 GOSUB 5600: PRINT@2,21;M1$;@4,21;M2$:USR(62)
6350 GET M$ : IF M$=" " GOTO 6350: REM*GET MOVE*
6360 IF M$="T" THEN P=X-1 : Q=Y-1 : GOTO 6450
6370 IF M$="Y" THEN P=X-1 : Q=Y : GOTO 6450
6380 IF M$="U" THEN P=X-1 : Q=Y+1 : GOTO 6450
6390 IF M$="G" THEN P=X : Q=Y-1 : GOTO 6450
6400 IF M$="J" THEN P=X : Q=Y+1 : GOTO 6450
6410 IF M$="B" THEN P=X+1 : Q=Y-1 : GOTO 6450
6420 IF M$="N" THEN P=X+1 : Q=Y : GOTO 6450
6430 IF M$="M" THEN P=X+1 : Q=Q+1 : GOTO 6450
6440 GOTO 6350
6450 IF P>16 THEN USR(62):GOTO 6340
6460 N=SU(P,Q):IF N<>1 GOTO 6490
6470 GOSUB 5600: REM*CLEAR RT/HAND SIDE OF SCREEN*
6480 PRINT@2,21;M3$;@4,21;M4$ :GOSUB 5570: GOTO 6340
6490 PRINT@X,Y;" ";@P,Q;P$
6500 ST=ST+1 : MO(ST,0)=P : MO(ST,1)=Q
6510 IF N>1 GOTO 6530
6520 X=P : Y=Q : GOTO 6330
6530 NS=N-1
6540 IF NS<8 THEN SA=NS : GOTO 6560
6550 IF NS>7 THEN SB=NS-7 : GOTO 6570
6560 ON SA GOSUB 1640,1980,2220,2550,3060,3470,3790: GOTO 6580
6570 ON SB GOSUB 4080,4330,4560,4690,4850,5110,5370
6580 IF ST=250 GOTO 6140
6590 IF VT=<0 GOTO 6140: REM*NO VITALITY*
6600 GOTO 6330
6610 DATA 1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1
6620 DATA 1,1,0,0,0,2,0,0,0,1,0,0,0,2,0,0,0,1,1
6630 DATA 1,1,0,0,0,1,0,0,0,1,0,0,0,1,0,0,0,1,1
6640 DATA 1,1,0,0,0,1,0,0,0,1,0,0,0,1,0,0,0,1,1
6650 DATA 1,1,2,1,1,1,1,1,2,1,2,1,1,1,1,1,2,1,1
6660 DATA 1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1
6670 DATA 1,2,1,1,1,1,1,1,1,0,1,1,1,1,1,2,1,1
6680 DATA 1,0,0,0,1,0,0,0,2,0,2,0,0,0,0,0,0,0,1
6690 DATA 1,0,0,0,1,0,0,0,1,0,1,0,0,0,1,0,0,0,1
6700 DATA 1,0,0,0,2,0,0,0,1,0,1,0,0,0,1,0,0,0,1
6710 DATA 1,1,2,1,1,2,1,1,1,0,1,1,1,2,1,2,1,1,1
6720 DATA 1,0,0,0,1,0,0,0,1,0,1,0,0,0,1,0,0,0,1
6730 DATA 1,0,0,0,2,0,0,0,1,0,1,0,0,0,1,0,0,0,1
6740 DATA 1,0,0,0,1,0,0,0,2,0,2,0,0,0,2,0,0,0,1
6750 DATA 1,1,1,1,1,1,1,1,1,1,0,1,1,1,1,1,1,1,1
6760 DATA 1,1,1,1,1,1,1,1,1,1,0,1,1,1,1,1,1,1,1
6770 DATA 2,3,2,7,2,11,2,15,8,2,8,6,8,12,8,16
6780 DATA 12,2,12,6,12,12,12,16
```

```
 0.%            WANDER              %
 1.%        by  P WILLIAMS          %
 2.%            JAN 1982            %
 3.VAR BOARD:ARRAY[11,11]OF CHAR;
 4.    X,Y,DIRECTION:INTEGER;
 5.    A,PLAYER:CHAR;
 6.    GAME:BOOLEAN;
 7.PROCEDURE INSTRUCT;
 8.BEGIN
 9.   WRITELN("■WANDER");
10.   WRITELN(" ──────");
11.   WRITELN("THIS IS A GAME FOR TWO PLAYERS");
12.   WRITELN();
13.   WRITELN("EACH MOVES ON THE BOARD ALTERNATELY");
14.   WRITELN();
15.   WRITELN("TO CREATE A LINE THAT WANDERS ROUND");
16.   WRITELN();
17.   WRITELN("TO WIN THE PLAYER MUST EITHER REACH");
18.   WRITELN();
19.   WRITELN("THE HOME SQUARE,MARKED WITH A LETTER");
20.   WRITELN();
21.   WRITELN("OR MAKE THE OPPONENT GO OFF THE EDGE");
22.   WRITELN();
23.   WRITELN("OF THE BOARD. THERE ARE 4 DIRECTIONS ");
24.   WRITELN();
25.   WRITELN("TO MOVE,L FOR LEFT,R FOR RIGHT");
26.   WRITELN();
27.   WRITELN("U FOR UP AND D FOR DOWN");
28.   WRITELN();
29.   WRITELN("PRESS CR  KEY TO CONTINUE");
30.   WRITELN("──────────────────────────────");
31.   READ(A);
32.   WRITELN("■ PLEASE NOTE THAT THE LINE WILL NOT ");
33.   WRITELN();
34.   WRITELN("TURN BACK ON ITSELF");
35.   WRITELN();
36.   WRITELN("PRESS CR KEY TO CONTINUE");
37.   WRITELN("──────────────────────────");
38.   READ(A);
39.END;
40.PROCEDURE BOARDSHOW;%      DRAWS THE BOARD %
41.   VAR I,J:INTEGER;
42.BEGIN
43.   WRITE("■");
44.   MOVE(0,2);
45.   FOR I:=1 TO 2 DO BEGIN
46.     FOR J:=1 TO 22 DO BEGIN
47.       WRITE("*")
48.     END;
49.     WRITELN();
50.   END;
51.   FOR I:=1 TO 9 DO BEGIN
52.     WRITELN("**┌─┬─┬─┬─┬─┬─┬─┬─┬─┬─┐**");
53.     WRITELN("**└─┴─┴─┴─┴─┴─┴─┴─┴─┴─┘**")
54.   END;
55.   FOR I:=1 TO 2 DO BEGIN
56.     FOR J:=1 TO 22 DO BEGIN
57.       WRITE("*")
58.     END;
59.     WRITELN()
```

```
60.   END;
61.   MOVE(10,3);
62.   WRITE("▼");
63.   MOVE(2,20);
64.   WRITE("■■■");
65.   MOVE(2,21);
66.   WRITE("■■");
67.   MOVE(18,20);
68.   WRITE("■■■");
69.   MOVE(18,21);
70.   WRITE("■B");
71.END;
72.PROCEDURE MOVE(X,Y:INTEGER);% THIS GOES TO PLACE ON BOARD %
73.   VAR RIGHT,DOWN:INTEGER;
74.BEGIN
75.   WRITE("■");
76.   FOR RIGHT:=1 TO X DO BEGIN
77.     WRITE("■")
78.   END;
79.   FOR DOWN:=1 TO Y DO BEGIN
80.     WRITE("■")
81.   END;
82.END;
83.PROCEDURE BOARDSET;% LOADS START POSITION %
84. VAR I,J:INTEGER;
85.BEGIN
86.   FOR I:=1 TO 11 DO BEGIN
87.     FOR J:=1 TO 11 DO BEGIN
88.       BOARD[I,J]:=' '
89.     END
90.   END;
91.   FOR I:=1 TO 11 DO BEGIN
92.     BOARD[1,I]:='*';
93.     BOARD[11,I]:='*';
94.     BOARD[I,1]:='*';
95.     BOARD[I,11]:='*';
96.   END;
97.   BOARD[2,10]:='■';
98.   BOARD[10,10]:='■';
99.END;
100.PROCEDURE PLOT;% ASKS FOR MOVE & SHOWS ON THE BOARD %
101.VAR GO:CHAR;NGO,ND:INTEGER;
102.BEGIN
103.   IF BOARD[X,Y]=' 'THEN
104.   BEGIN
105.     REPEAT
106.         MOVE(25,2);
107.         WRITE("PLAYER ",PLAYER:1,"'S TURN");
108.         MOVE(25,4);
109.         WRITE("L OR R  ? ");
110.         MOVE(25,6);
111.         WRITELN("U OR D  ?");
112.         MOVE(25,8);
113.         WRITELN("PRESS UNTIL ");
114.         MOVE(25,10);
115.         WRITELN("I  BEEP");
116.         BLINK;
117.         GO:=KEY;
118.         CASE GO OF 'L':NGO:=2;
119.                    'R':NGO:=3;
```

```
120.                    'U':NGO:=1;
121.                    'D':NGO:=4
122.        END;
123.    UNTIL((GO='L')OR(GO='R')OR(GO='U')OR(GO='D'))AND
124.    (DIRECTION+NGO<>5);
125.    CALL(62)ND,ND;
126.    MOVE(2*X-2,Y*2);
127.    IF NGO=DIRECTION THEN
128.    BEGIN
129.      WRITE("⊥■■■ | ");
130.      BOARD[X,Y]:='+'
131.    END
132.    ELSE
133.      BEGIN
134.        CASE(NGO+DIRECTION)OF 3:BOARD[X,Y]:='\';
135.                              7:BOARD[X,Y]:='\';
136.                              4:BOARD[X,Y]:='/';
137.                              6:BOARD[X,Y]:='/'
138.        END;
139.        IF(NGO+DIRECTION=3)OR(NGO+DIRECTION=7)THEN WRITE(" \■■■\ ");
140.        IF(NGO+DIRECTION=4)OR(NGO+DIRECTION=6)THEN WRITE("/ ■■■ /")
141.      END;
142.      IF PLAYER='A'THEN PLAYER:='B'ELSE PLAYER:='A';
143.    UPDATE(NGO);
144.    DIRECTION:=NGO;
145.    PLOT
146.  END
147.  ELSE
148.  BEGIN
149.       IF(BOARD[X,Y]='*')
150.       OR(BOARD[X,Y]='▒')
151.       OR(BOARD[X,Y]='▓')
152.        THEN
153.        BEGIN
154.        IF BOARD[X,Y]='▒'THEN PLAYER:='A';
155.        IF BOARD[X,Y]='▓'THEN PLAYER:='B';
156.        END
157.        ELSE
158.          BEGIN
159.            IF BOARD[X,Y]='/'THEN
160.              CASE DIRECTION OF 1:ND:=3;
161.                                4:ND:=2;
162.                                2:ND:=4;
163.                                3:ND:=1
164.              END
165.            ELSE IF BOARD[X,Y]='\'THEN
166.              CASE DIRECTION OF 1:ND:=2;
167.                                4:ND:=3;
168.                                2:ND:=1;
169.                                3:ND:=4
170.              END
171.            ELSE ND:=DIRECTION;
172.            DIRECTION:=ND;
173.            UPDATE(ND);
174.            PLOT
175.          END;
176.  END;
177.END;
178.PROCEDURE BLINK;% CURSOR ON NEXT SQURE %
179.VAR Z:INTEGER;
```

```
180.BEGIN
181.  MOVE(2*X-2,2*Y);
182.  WRITE("┴▒▒▒┤ ");
183.  Z:=1;
184.  WHILE Z>0 DO Z:=Z-1;
185.  MOVE(2*X-2,2*Y);
186.  WRITE("  ▒▒▒  ");
187.  Z:=1;
188.  WHILE Z>0 DO Z:=Z-1;
189.  MOVE(2*X-2,2*Y);
190.  WRITE("^▒▒▒v")
191.END;
192.PROCEDURE UPDATE(Q:INTEGER);% POINTS TO NEXT SQUARE %
193.BEGIN
194.  CASE Q OF 1:Y:=Y-1;
195.            4:Y:=Y+1;
196.            2:X:=X-1;
197.            3:X:=X+1
198.  END;
199.END;
200.%                    MAIN PART OF PROGRAM        %
201.BEGIN
202.  INSTRUCT;
203.  GAME:=TRUE;
204.  WHILE GAME=TRUE DO
205.  BEGIN
206.    X:=6;
207.    Y:=2;
208.    PLAYER:='A';
209.    DIRECTION:=4;
210.    BOARDSHOW;
211.    BOARDSET;
212.    PLOT;
213.    WRITE("▒");
214.    MOVE(10,5);
215.    WRITE("PLAYER ",PLAYER:1," IS THE WINNER");
216.   WRITELN();
217.   WRITELN();
218.   WRITELN("WANT TO PLAY AGAIN? Y/N  ");
219.   READ(A);
220.   IF A='N'THEN GAME:=FALSE
221. END
222.END.
223.
```

```
50 REM*WRITTEN IN SHARP EXTENDED BASIC*
60 REM*BY F.E.WOODWARD,8 VIOLET AV.RAMSGATE*
70 REM*INCLUDES A USEFUL CARD SHUFFLE AND DEAL ROUTINE (SUB 5000)*
80 REM*ALSO EXTENSIVE ERROR TRAPPING ROUTINES*
90 CLR:RESTORE
100 REM*INITIALISATION OF VARIABLES*
110 DIM RO(89):DIM DP(89,2):DIM MA$(52,2):DIM D(4,13):DIM CD$(52)
120 B$="▒":MB$="▒▒▒"
130 M1$="CARD 1="
140 M2$="CARD 2="
150 SP$=SPC(16)
290 PRINT"▒"
300 PRINT@4,3;"▌ ▌ ▛▀ ▌  ▌  ▗▄▖ "
310 PRINT@5,3;"▙▟ ▙ ▌  ▌  ▌  ▌"
320 PRINT@6,3;"▌ ▌ ▙▄ ▙▄ ▙▄ ▝▄▘ "
340 PRINT@4,21;"▀▀▀ ▌ ▌ ▛▀ ▛▀▜ ▛▀ "
350 PRINT@5,21;" ▌  ▙▟ ▙  ▙▄▟ ▙  "
360 PRINT@6,21;" ▌  ▌ ▌ ▙▄ ▌ ▌ ▙▄ "
370 PRINT@11,2;"WELCOME TO THE MZ-80K COMPUTER GAME"
380 PRINT@15,2;"DO'NT GO AWAY--THERE WILL BE A SHORT"
390 PRINT@18,2;"PAUSE WHILE I PRIME MY MEMORY MATRIX"
400 GOTO 10000
1000 REM*OPENING DISPLAY ROUTINE*
1010 FOR I=1 TO 89
1020 Z=INT(89*RND(1)+1)
1030 IF RO(Z)>0 GOTO 1020
1040 RO(Z)=I
1050 NEXT I
1055 PRINT"▒"
1060 FOR Y=4 TO 35:PRINT@5,Y;B$;@15,Y;B$:NEXT Y
1070 FOR X=5 TO 15:PRINT@X,4;B$;@X,35;B$:NEXT X
1080 FOR P=1 TO 89
1090 N=RO(P)
1100 IF DP(N,2)=1 THEN D$="▚":GOTO 1160
1110 IF DP(N,2)=2 THEN D$="▒":GOTO 1160
1120 IF DP(N,2)=3 THEN D$="▞":GOTO 1160
1130 IF DP(N,2)=4 THEN D$="▜":GOTO 1160
1140 IF DP(N,2)=5 THEN D$="▚":GOTO 1160
1160 PRINT@DP(N,0),DP(N,1);D$
1170 MU=INT(10*RND(1)+1)
1180 POKE 4514,MU:USR(68):USR(71)
1190 NEXT P
1200 PRINT@18,3;"A GAME FOR TWO PLAYERS WHO HAVE TO"
1210 PRINT@20,4;"MATCH PAIRS OF CARDS OF THE SAME"
1220 PRINT@22,15;"VALUE."
1230 PP=3:GOSUB 8000
1235 PRINT"▒":S1=0:S2=0
1240 PRINT@3,3;"EACH PLAYER WILL BE ASKED IN TURN TO"
1250 PRINT@5,3;"GIVE THE POSITION OF TWO CARDS ON THE"
1260 PRINT@7,3;"DISPLAY MATRIX BY KEYING IN THE CARDS"
1270 PRINT@9,3;"COORDINATES (eg.8,7) FOLLOWED BY THE"
1280 PRINT@11,3;"'CR'KEY."
1290 PRINT@13,3;"IF THE TWO CARDS MATCH IN VALUE A"
1300 PRINT@15,3;"POINT IS SCORED AND THAT PLAYER HAS"
1310 PRINT@17,3;"ANOTHER TURN."
1320 PRINT@19,3;"AT THE END OF EACH ROUND THE CARDS"
1330 PRINT@21,3;"ARE SHUFFLED AND RE-DEALT.HAVE FUN."
1350 REM*SHUFFLE PACK &SET GAME MATRIX*
1360 GOSUB 5000
1370 PRINT"▒":TT=0
1380 FOR M=1 TO 52
1390 X=VAL(MA$(M,0)):Y=VAL(MA$(M,1))
1400 PRINT@X,Y;MB$
1410 NEXT M
```

```
1420 PRINT@1,5;"1";@9;"2";@13;"3";@17;"4";@21;"5";@25;"6";@29;"7";@33;"8"
1430 PRINT@1,37;"9";@3,2;"A";@5,2;"B";@7,2;"C";@9,2;"D";@11,2;"E";@13,2;"F"
1440 PRINT@15,2;"PLAYER 1";@20;"PLAYER 2"
1450 PRINT@16,2;"--------";@20;"--------"
1460 PRINT@21,2;"SCORE=";@20;"SCORE="
1470 PRINT@21,12;S1;@30;S2
1500 REM*1st.PLAYERS ROUTINE*
1510 PRINT@17,2;M1$;:INPUT CL$,CN
1511 GOSUB 4000
1512 IF ER=0 GOTO 1520
1513 PRINT@17,2;SP$
1514 GOSUB 4500
1515 GOTO 1510
1520 GOSUB 3000
1530 C1=CN
1540 X=VAL(MA$(C1,0)):Y=VAL(MA$(C1,1))
1550 C1$=MA$(C1,2)
1551 IF C1$<>"    " GOTO 1560
1552 GOSUB4500
1553 PRINT@17,2;SP$:GOTO 1510
1560 PRINT@X,Y;C1$
1570 PRINT@19,2;M2$;:INPUT CL$,CN
1571 GOSUB 4000
1572 IF ER=0 GOTO 1580
1573 PRINT@19,2;SP$
1574 GOSUB 4500
1575 GOTO 1560
1580 GOSUB 3000
1590 C2=CN
1600 P=VAL(MA$(C2,0)):Q=VAL(MA$(C2,1))
1610 C2$=MA$(C2,2)
1611 IF C2$<>"    " GOTO 1620
1612 GOSUB 4500
1613 PRINT@19,2;SP$:GOTO 1570
1620 PRINT@P,Q;C2$
1630 PP=2:GOSUB 8000 :REM*PAUSE*
1640 PRINT@17,2;SP$;@19,2;SP$
1650 IF LEFT$(C1$,2)=LEFT$(C2$,2) GOTO 1680
1660 PRINT@X,Y;MB$;@P,Q;MB$
1670 GOTO 1800:REM*OTHER PLAYER*
1680 S1=S1+1:TT=TT+1:USR(62):PRINT@21,12;S1
1683 MA$(C1,2)="    ":MA$(C2,2)="    "
1685 PRINT@X,Y;SPC(3):PRINT@P,Q;SPC(3)
1690 IF TT=26 GOTO 2510
1700 GOTO 1510 : REM*REPEAT TURN*
1800 REM*2nd.PLAYERS ROUTINE*
1810 PRINT@17,20;M1$;:INPUT CL$,CN
1811 GOSUB 4000
1812 IF ER=0 GOTO 1820
1813 PRINT@17,20;SP$
1814 GOSUB 4500
1815 GOTO 1810
1820 GOSUB 3000
1830 D1=CN
1840 X=VAL(MA$(D1,0)):Y=VAL(MA$(D1,1))
1850 D1$=MA$(D1,2)
1851 IF D1$<>"    " GOTO 1860
1852 GOSUB 4500
1853 PRINT@17,20;SP$:GOTO 1810
1860 PRINT@X,Y;D1$
1870 PRINT@19,20;M2$;:INPUT CL$,CN
1871 GOSUB 4000
1872 IF ER=0 GOTO 1880
1873 PRINT@19,20;SP$
```

```
1874 GOSUB 4500
1875 GOTO 1870
1880 GOSUB 3000
1890 D2=CN
1900 P=VAL(MA$(D2,0)):Q=VAL(MA$(D2,1))
1910 D2$=MA$(D2,2)
1911 IF D2$<>"    " GOTO 1920
1912 GOSUB 4500
1913 PRINT@19,20;SP$:GOTO 1870
1920 PRINT@P,Q;D2$
1930 PP=2:GOSUB 8000 :REM*PAUSE*
1940 PRINT@17,20;SP$;@19,20;SP$
1950 IF LEFT$(D1$,2)=LEFT$(D2$,2) GOTO 1980
1960 PRINT@X,Y;MB$;@P,Q;MB$
1970 GOTO 1500
1980 S2=S2+1:TT=TT+1:USR(62):PRINT@21,30;S2
1983 MA$(D1,2)="    ":MA$(D2,2)="    "
1985 PRINT@X,Y;SPC(3):PRINT@P,Q;SPC(3)
1990 IF TT=26 GOTO 2510
2000 GOTO 1810 : REM*REPEAT TURN*
2500 REM*GAME OVER ROUTINE*
2510 PRINT"@":PRINT@2,10:"ROUND OVER"
2520 PRINT@4,6;"1st.PLAYERS SCORE NOW =";@34;S1
2530 PRINT@8,6;"2nd.PLAYERS SCORE NOW =";@34;S2
2540 PRINT@12,2;"DO YOU WISH TO PLAY A FURTHER ROUND"
2550 PRINT@14,8;"KEY Y or N"
2560 GET AN$
2570 IF AN$="Y" THEN PRINT"@":PRINT@6,10;"JUST LET ME RE-DEAL":GOTO 1350
2580 IF AN$="N" GOTO 2600
2590 GOTO 2560
2600 PRINT@18,2;"DO YOU WANT A NEW GAME THEN? (Y or N)
2610 GET BN$
2620 IF BN$="Y" GOTO 1235
2630 IF BN$="N" THEN PRINT"@":GOTO 2650
2640 GOTO 2610
2650 PRINT@10,4;"SO-LONG FOR NOW THEN"
2660 END
3000 REM*CO-ORD.DECODING ROUTINE*
3010 IF CL$="A" THEN CN=CN+0:GOTO 3070
3020 IF CL$="B" THEN CN=CN+9:GOTO 3070
3030 IF CL$="C" THEN CN=CN+18:GOTO 3070
3040 IF CL$="D" THEN CN=CN+27:GOTO 3070
3050 IF CL$="E" THEN CN=CN+36:GOTO 3070
3060 IF CL$="F" THEN CN=CN+44:GOTO 3070
3070 RETURN
4000 REM*ERROR TRAP ROUTINE*
4010 CL$=RIGHT$(CL$,1):ER=0
4020 IF (CN<1)+(CN>9) GOTO 4070
4030 IF (ASC(CL$)<65)+(ASC(CL$)>70) GOTO 4070
4040 IF (ASC(CL$)=70)*(CN=1) GOTO 4070
4050 IF (ASC(CL$)=70)*(CN=9) GOTO 4070
4060 RETURN
4070 ER=1
4080 RETURN
4500 REM*INVALID ENTRY SOUND EFFECT*
4510 FOR V=15 TO 20:POKE 4514,V:USR(68):NEXT V
4520 FOR V=20 TO 15 STEP -1:POKE 4514,V:USR(68):NEXT V:USR(71)
4530 RETURN
5000 REM*CARSHUFFLE AND DEAL ROUTINE*
5010 FOR J=1 TO 4
5020 FOR K=1 TO13
5030 D(J,K)=0
5040 NEXT K
5050 NEXT J
```

```
5060 FOR Z=1 TO 52
5070 S=INT(4*RND(1)+1)
5080 N=INT(13*RND(1)+1)
5090 IF D(S,N)=-1 GOTO 5070
5100 D(S,N)=-1
5110 ON S GOTO 5120,5130,5140,5150
5120 S$="♠":GOTO 5160
5130 S$="♥":GOTO 5160
5140 S$="♦":GOTO 5160
5150 S$="♣":GOTO 5160
5160 IF N=1 THEN F$="A":GOTO 5220
5170 IF N>10 THEN F=N-10:ON F GOTO 5180,5190,5200
5175 GOTO 5210
5180 F$="J":GOTO 5220
5190 F$="Q":GOTO 5220
5200 F$="K":GOTO 5220
5210 F$=STR$(N)
5220 C$=F$+S$
5230 L=LEN(C$)
5240 IF L=2 THEN C$=" "+C$
5250 CD$(Z)=C$
5260 NEXT Z
5270 FOR P=1 TO 52:MA$(P,2)=CD$(P):NEXT P
5280 RETURN
8000 REM*PAUSE ROUTINE*
8010 FOR T=1 TO PP*1000:NEXT T
8020 RETURN
10000 REM*FILL OPENING DISPLAY MATRIX*
10010 FOR Z=1 TO 89:READ DP(Z,0),DP(Z,1),DP(Z,2):NEXT Z
10020 DATA 7,6,1,7,7,2,7,8,2,7,9,2,7,10,5,7,13,3,7,14,2,7,15,2,7,16,5
10030 DATA 7,19,1,7,20,2,7,21,4,7,23,1,7,24,2,7,25,2,7,26,2,7,27,5
10040 DATA 7,30,3,7,31,2,7,32,2,7,33,5,8,7,2,8,10,2,8,13,2,8,16,2,8,20,2
10050 DATA 8,24,2,8,27,2,8,30,2,8,33,1,9,7,2,9,10,2,9,13,2,9,16,2,9,20,2
10060 DATA 9,24,2,9,27,2,9,30,2,10,7,2,10,8,2,10,9,2,10,10,4,10,13,2
10070 DATA 10,14,2,10,15,2,10,16,2,10,20,2,10,24,2,10,25,2,10,26,2,10,27,
10080 DATA 10,30,1,10,31,2,10,32,2,10,33,5,11,7,2,11,13,2,11,16,2,11,20,2
10090 DATA 11,24,2,11,26,1,11,27,5,11,33,2,12,7,2,12,13,2,12,16,2,12,20,2
10100 DATA 12,24,2,12,27,2,12,30,5,12,33,2,13,6,3,13,7,2,13,8,5,13,12,3
10110 DATA 13,13,2,13,16,2,13,17,5,13,19,3,13,20,2,13,21,5,13,23,3,13,24,
10120 DATA 13,27,2,13,28,5,13,30,1,13,31,2,13,32,2,13,33,4
10130 REM*FILL GAME DISPLAY MATRIX*
10140 FOR R=1 TO 52:READ MA$(R,0),MA$(R,1):NEXT R
10150 DATA 3,4,3,8,3,12,3,16,3,20,3,24,3,28,3,32,3,36
10160 DATA 5,4,5,8,5,12,5,16,5,20,5,24,5,28,5,32,5,36
10170 DATA 7,4,7,8,7,12,7,16,7,20,7,24,7,28,7,32,7,36
10180 DATA 9,4,9,8,9,12,9,16,9,20,9,24,9,28,9,32,9,36
10190 DATA 11,4,11,8,11,12,11,16,11,20,11,24,11,28,11,32,11,36
10200 DATA 13,8,13,12,13,16,13,20,13,24,13,28,13,32
10210 GOTO 1000
```

```
%┌──────────────────┐%
% Pascal's triangle %
% ■ J.R.BROWN      %
% Date 9.4.1982    %
%└──────────────────┘%
%%
VAR X,NUMBER,SPACES:INTEGER;
    TRI:ARRAY[11]OF INTEGER;
PROCEDURE CLEAR;
  BEGIN
    FOR X:=2 TO 11 DO TRI[X]:=0;
    TRI[1]:=1;
  END;
PROCEDURE INPNUM;
  VAR B:CHAR;
  BEGIN
    WRITE("■■Enter N for the expansion [1+X]■N■ ? ");
    B:=KEY;
    WHILE(B<'0')OR(B>'9')DO B:=KEY;
    NUMBER:=ORD(B)-47;
    WRITELN(B:1,"■");
  END;
PROCEDURE PRINT;
  BEGIN
    FOR X:=1 TO SPACES DO WRITE(" ");
    X:=1;
    WHILE TRI[X]<>0 DO
      BEGIN
        WRITE(TRI[X]:4);
        X:=X+1;
      END;
    WRITELN();
  END;
PROCEDURE NEXT;
  BEGIN
    FOR X:=10 DOWNTO 2 DO
      TRI[X]:=TRI[X]+TRI[X-1];
  END;
% MAIN PROGRAM %
BEGIN
  CLEAR;
  INPNUM;
  SPACES:=20;
  REPEAT
    SPACES:=SPACES-2;
    PRINT;
    NEXT;
    NUMBER:=NUMBER-1;
  UNTIL NUMBER=0;
END.
%┌──────────────────┐%
% Five Tiles       %
% ■ J.R.Brown      %
% Date 10.4.82     %
%└──────────────────┘%
%%
VAR X,Y,POSN,SCORE,TILE1,TILE2:INTEGER;
    MOVE,CONT,DR:CHAR;
    FLAG:BOOLEAN;
PROCEDURE TAB(N:INTEGER);
```

P A S C A L

Our thanks to Mr. T.J.P. Williams and Mr. J.T. Brown for the following PASCAL programs. If you have written any PASCAL programs or would like to pass on tips to other members, then send them to us for inclusion in future issues.

```
   BEGIN POKE(CHR(N),4465)END;
 PROCEDURE CURSOR(X,Y:INTEGER);
   BEGIN
     POKE(CHR(X),4465);
     POKE(CHR(Y),4466);
   END;
 PROCEDURE TITLES;
   BEGIN
     WRITE("■");TAB(10);
     WRITELN("F I V E  T I L E S");
     WRITE("■Try to stop the computer setting five  tiles");
     WRITELN(" in a pile at the bottom of the  screen.");
     WRITELN("■For each tile stopped you get 5 points.");
     WRITELN("■To control the '▬▬' at the centre of  the screen.");
     WRITELN("■Press 'A' once to move left.");
     WRITELN("Press 'f' once to move right.");
     WRITELN("■Any other key makes you stand still.");
     WRITELN("■The game is over when there are 5 tiles");
     WRITELN("in a pile.");
     WRITE("■     Press any key to continue");
     MOVE:=KEY;
     WHILE ORD(MOVE)=0 DO MOVE:=KEY;
   END;
 PROCEDURE SETSCREEN;
   BEGIN
     WRITE("■");
     FOR X:=-11408 TO-11289 DO
       POKE(CHR(163),X);
     MOVE:=' ';
     POSN:=20;
   END;
 PROCEDURE MOVETILE;
   BEGIN
     IF Y=0 THEN Y:=TRUNC(RND(1.0)*41.0)-12248;
     CONT:=PEEK(Y+40);
     IF(ORD(CONT)=166)AND(Y<-11568)THEN FLAG:=TRUE;
     IF ORD(CONT)=122 THEN
         BEGIN
           POKE(CHR(0),Y);
           Y:=0;
           SCORE:=SCORE+5;
           WRITE("■Score:",SCORE:3);
         END
     ELSE
       IF(ORD(CONT)=166)OR(Y>-11449)THEN Y:=0
     ELSE
       BEGIN
         POKE(CHR(0),Y);
         Y:=Y+40;
         POKE(CHR(166),Y);
       END;
   END;
 % MAIN PROGRAM %
 BEGIN
   TITLES;SETSCREEN;FLAG:=FALSE;
   SCORE:=0;TILE1:=0;TILE2:=0;
   REPEAT
     DR:=KEY;
     IF ORD(DR)=0 THEN DR:=MOVE;
     MOVE:=DR;
```

```
    IF MOVE='A'THEN POSN:=POSN-1;
    IF POSN=-1 THEN POSN:=0;
    IF MOVE='L'THEN POSN:=POSN+1;
    IF POSN=37 THEN POSN:=36;
    CURSOR(POSN,16);
    IF(POSN>0)AND(POSN<37)THEN
          WRITE("■      ■■■■■");
    WRITE("■■■■■ ");
    Y:=TILE1;MOVETILE;TILE1:=Y;
    Y:=TILE2;MOVETILE;TILE2:=Y;
  UNTIL FLAG;
  WRITE("■Score:",SCORE:3);
  CURSOR(12,9);
  WRITE("G O O D  B Y E");
END.
%┌──────────┐%
%│  Racer   │%
%│■ J.R.Brown│%
%│Date 11.4.82│%
%└──────────┘%
%%
VAR LM,Z,L,K,G,R,X,I,C,C1,V:INTEGER;
    MOVE:CHAR;
    W:REAL;
    FLAG:BOOLEAN;
PROCEDURE TITLES;
  BEGIN
    WRITE("■The object of the game is to steer a caralong a road as far")
    WRITELN(" as possible.");
    WRITELN("■Controls are 'A' for left.");
    WRITELN("              'D' for right.");
    WRITELN("■Any other key for straight ahead.");
    WRITELN("■Press any key to continue");
    MOVE:=KEY;
    WHILE ORD(MOVE)=0 DO MOVE:=KEY;
  END;
PROCEDURE SETSCREEN;
  BEGIN
    WRITE("■");
    FOR I:=1 TO 20 DO WRITELN("          ■■■     ■■■");
    I:=10;C:=15;C1:=C;V:=-11848;
    X:=V+40;W:=1.0/14.0;L:=0;
    R:=0;LM:=0;
  END;
BEGIN
  TITLES;
  SETSCREEN;
  FLAG:=FALSE;
  REPEAT
    K:=TRUNC(RND(1.0)*9.0);
    MOVE:=KEY;
    G:=ORD(MOVE);
    IF G=0 THEN G:=LM;
    IF G=65 THEN C:=C-1;
    IF G=68 THEN C:=C+1;
    LM:=G;
    IF(K=3)AND(L>=0)THEN L:=L-1
      ELSE
        IF(K=2)AND(L<=0)THEN L:=L+1;
    IF(I=0)OR(I=28)THEN
```

```
        BEGIN
          L:=0;
          I:=I+1-TRUNC(FLOAT(I)*W);
        END;
      I:=I+L;
      POKE(CHR(I),4465);
      WRITELN("▒▒▒      ▒▒▒");
      IF ORD(PEEK(X+C))=67 THEN
          FLAG:=TRUE
        ELSE
          BEGIN
            POKE(CHR(0),V+C1);
            POKE(CHR(201),X+C);
            C1:=C;
            R:=R+1;
          END;
      FOR Z:=1 TO 50 DO WRITE();
    UNTIL FLAG;
    WRITE("▒");POKE(CHR(12),4465);
    WRITELN("C R A S H");
    WRITELN("▒▒You travelled",R:5," miles.");
  END.
  %┌──────────┐%
  %│  MUTANTS │%
  %│▒ J.R.Brown│%
  %│Date 19.4.82│%
  %└──────────┘%
  %%
  VAR BOMBS,Z,SC,M,X,POS:INTEGER;
      Y:REAL;
      LM,DR:CHAR;
      FLAG,ZAP:BOOLEAN;
  PROCEDURE TITLES;
    BEGIN
      WRITELN("▒ It is the year 3002 and the earth has");
      WRITELN("been invaded by 100 mutants '",CHR(99):1,"'.");
      WRITELN("▒ Your mission is to destroy them all by");
      WRITELN("running over as many as possible before being blown up.");
      WRITELN("▒ They defend themselves by throwing out");
      WRITELN("mines 'O'.");
      WRITELN("▒ Your protection is 5 plasma bombs");
      WRITELN("which are detonated by pressing 'SPACE'");
      WRITELN("▒ The bombs are only effective within 2");
      WRITELN("quadrants around you.");
      WRITELN("▒          W");
      WRITELN("          ↑");
      WRITELN("      A ←+→ D      ARE YOUR CONTROLS");
      WRITELN("          ↓");
      WRITELN("          X");
      WRITE("▒Press any key to start");
      DR:=KEY;
      WHILE ORD(DR)=0 DO DR:=KEY;
    END;
  PROCEDURE SETSCREEN;
    BEGIN
      WRITE("▒");
      POS:=-12248;
      FOR X:=1 TO 100 DO
        BEGIN
          Y:=RND(1.0)*960.0;
```

```
        POKE(CHR(202),TRUNC(Y)+POS);
      END;
    POS:=POS+460;FLAG:=FALSE;SC:=0;
    LM:=' ';BOMBS:=5;ZAP:=FALSE;
  END;
PROCEDURE MOVE;
  VAR TP:INTEGER;
  BEGIN
    POKE(CHR(199),POS);
    Y:=RND(1.0);
    IF Y<0.1 THEN
        BEGIN
          TP:=0;
          WHILE ORD(PEEK(-12288+TP))<>0 DO TP:=TRUNC(RND(1.0)*960.0);
          POKE(CHR(74),-12288+TP);
        END;
    DR:=KEY;
    IF DR=' 'THEN ZAP:=TRUE
      ELSE
        BEGIN
          IF ORD(DR)=0 THEN DR:=LM;
          IF DR='W'THEN M:=-40;
          IF DR='A'THEN M:=-1;
          IF DR='D'THEN M:=+1;
          IF DR='X'THEN M:=+40;
          LM:=DR;
        END;
    POKE(CHR(0),POS);
    IF(POS+M<-12288)OR(POS+M)-11289)THEN M:=0;
    POS:=POS+M;
  END;
PROCEDURE SCORE;
  BEGIN
    IF ORD(PEEK(POS))=202 THEN SC:=SC+1;
    IF(ORD(PEEK(POS))=74)OR(SC=100)THEN FLAG:=TRUE;
    WRITE("Score (",SC:4,") Plasma bombs (",BOMBS:1,")");
  END;
PROCEDURE REMOVE;
  BEGIN
    IF BOMBS<>0 THEN
      BEGIN
        FOR Z:=POS-82 TO POS-78 DO
          IF ORD(PEEK(Z))=74 THEN POKE(CHR(0),Z);
        FOR Z:=POS-42 TO POS-38 DO
          IF ORD(PEEK(Z))=74 THEN POKE(CHR(0),Z);
        FOR Z:=POS-2 TO POS+2 DO
          IF ORD(PEEK(Z))=74 THEN POKE(CHR(0),Z);
        FOR Z:=POS+38 TO POS+42 DO
          IF ORD(PEEK(Z))=74 THEN POKE(CHR(0),Z);
        FOR Z:=POS+78 TO POS+82 DO
          IF ORD(PEEK(Z))=74 THEN POKE(CHR(0),Z);
        ZAP:=FALSE;
        BOMBS:=BOMBS-1;
      END;
  END;
BEGIN
  TITLES;
  SETSCREEN;
  REPEAT
    MOVE;
```

```
    IF ZAP THEN REMOVE;
    SCORE;
  JNTIL FLAG;
  WRITE("▒Score :",SC:3);
  IF SC<20 THEN WRITELN(" - Rating : CADET");
  IF(SC>19)AND(SC<50)THEN WRITELN(" - Rating : LIEUTENANT");
  IF(SC>49)AND(SC<99)THEN WRITELN(" - Rating : CAPTAIN");
  IF SC=100 THEN WRITELN(" - Rating : ADMIRAL ');
END.
```

BULLET PROOF LISTING

```
0 POKE10680,1:REM BK***" LIST:POKE6637,80:REM BK***" [SHIFT] & [BREAK]
5 PRINT"▒":C$="CODEWORD":TEMPO4
10 AA$="     SLOPPY SOFTWARE SECURITY SYSTEMS     "
15 AA$="     KEEP YOUR HANDS OFF THE KEYBOARD!     "+AA$
20 Z=LEN(AA$)
25 FORI=1TO3:FORN=1TOZ
30 BB$=RIGHT$(AA$,Z-N)+LEFT$(AA$,N)
35 PRINT"▒▒▒▒▒▒▒▒▒▒▒▒";LEFT$(BB$,40)
40 IFN/10=INT(N/10)THENPOKE4514,1:USR(68):USR(71)
45 IFN/5=INT(N/5)THENPOKE4514,1:USR(68):USR(71)
50 FORM=0TO50:NEXTM
70 NEXTN:NEXTI
75 FORN=1TO5:USR(62):NEXTN
80 PRINT"▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒ENTER CODE NOW!▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒.";
90 FORN=0TO7
100 POKE10167,1
110 GETA$:IFPEEK(17828)=0THEN110
120 PRINTA$;
130 IFPEEK(17828)<30THENBYE
131 IF(PEEK(17828)=>96)*(PEEK(17828)=<102)THENBYE
135 B$=B$+A$
140 USR(62):PRINT".";
150 NEXTN
160 IFC$=B$THEN190
170 BYE
180 END
190 PRINT"▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒ACCEPTED"
195 MUSIC"CODOEODOEOFOEOFOGOFOGOAOGOAOBOAOBO▒CO"
196 MUSIC"▒COBOAOBOAOGOAOGOFOGOFOEOFOEODOEODOCO"
200 POKE10680,0:POKE6637,30
```

SHARPSOFT Sharpsoft Ltd., 86-90 Paul Street, London EC2A 4NE
Printed by Oldham Press (T.U.), Chatham, Kent.