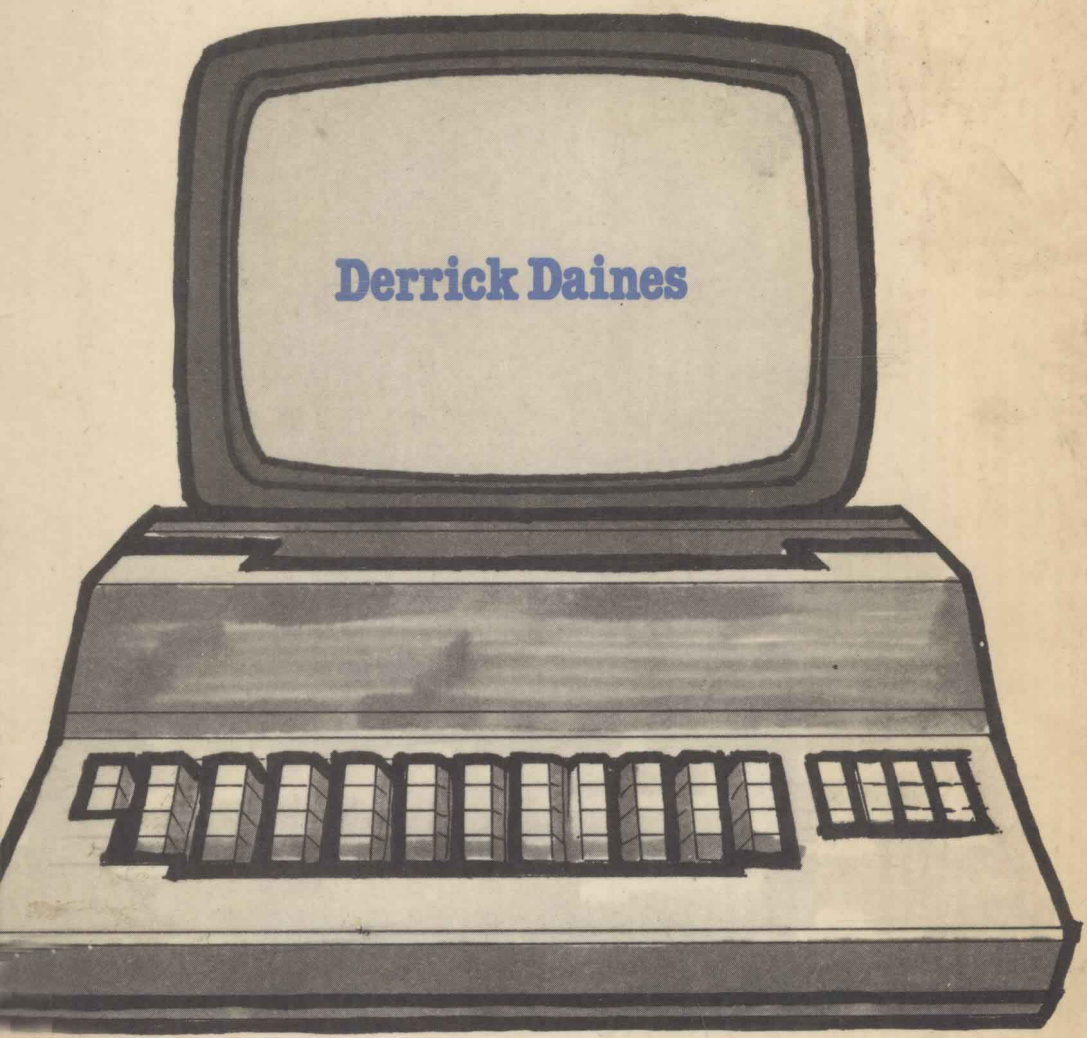
26 BASIC Programs for your Micro



Derrick Daines

Newnes Microcomputer Books

26 BASIC Programs for your Micro

Derrick Daines

Newnes Technical Books

Newnes Technical Books
is an imprint of the Butterworth Group
which has principal offices in
London, Sydney, Toronto, Wellington, Durban and Boston

First published 1982

British Library Cataloguing in Publication Data
Daines, Derrick
26 BASIC programs for your micro.
1. Microcomputers—Programming
2. Basic (Computer program language)
I. Title
001.64'2 QA76.6.

ISBN 0-408-01204-8

Typeset by Tunbridge Wells Typesetting Services Ltd.
Printed in England by Page Bros Ltd, Norwich Norfolk

Preface

Here is a book of unique computer programs — that is, programs that cannot be found anywhere else. Only two, 'Power boat' and 'Place value', have ever appeared in print before. This book, therefore, is not just another collection of well-worn programs but a wholly new text, with games and ideas that have never before seen the light of day. The programs are all my own, so if there is any blame to be attributed, that is mine too.

The programs can be graded in various ways. The method I have adopted here is an empirical one, being based on a subjective assessment of length plus complexity. No doubt this has given rise to quirks and anomalies, but it is hoped that these will not detract from the user's enjoyment.

The programs range from very simple to extremely complex. The former are capable of translation on to the simplest of computers that support BASIC as a programming language, while the latter require anything up to 13 K of memory. I have included in the notes to each program comments on any special requirements.

Also included are notes of interest to budding computer programmers. I have brought to the reader's attention any items of special interest in each program that are worthy of study and inclusion in other programs. In this way, I hope that the enjoyment I derive from computer programming may be passed on to others, and that the hobby we enjoy may be advanced. By studying these notes, the user will be guided towards better and more complex programs, using the techniques developed.

D.D.

Contents

Introduction

It tends to be axiomatic that the 'best' or 'most natural' version of BASIC is the one that happens to run on your machine. As the owner and user of many versions, I choose to write in SWTP BASIC because it is fairly easy to translate to other BASICs. All the programs in this book have been written in this version, utilising an SWTP 6800 computer with 32 K of memory, and saved on minifloppies. Very few commands are machine-dependent (I give a list of those that are; some may be omitted altogether). What follows here is a description of SWTP BASIC to smooth over any difficulties encountered.

VARIABLES A variable can be any single letter, or a letter and a number 0–9.

STRING VARIABLES These are a single letter followed by $, and are used to hold literal data. For example, A$ can be equated to "1224" or "Example", but not 1234. (The quotation marks define a string.) When SWTP BASIC initialises, strings are set to 32 characters maximum, but this may be extended by use of the STRING command, e.g. STRING = 50. Command lines are limited to 72 characters. Multiple statement lines are accepted with a colon (:) used as a separator.

CONCATENATION Strings may be concatenated (joined together) by use of the ' + ' symbol. For instance, A$ = "HELLO":B$ = "JOHN":C$ = A$ + B$ (C$ will now contain "HELLOJOHN")

ARRAYS If not dimensioned by a DIM statement, a 10-element array is assumed. Similarly with matrices — if not dimensioned, a 10 by 10 matrix is assumed. To save memory, therefore, it is wise to dimension first. String variables may also be dimensioned as an array.

INITIAL VALUES When variables are used for the first time, SWTP BASIC assumes that they have initial value 0. It is not necessary to assign 0 to them first, as in some BASICs.

LINE = X This command may be used to set the number of print positions in a line, where X is the desired number of positions. If the print position is within 25 per cent of the line length, and a space is encountered, SWTP BASIC will force a line-feed and carriage-return, thus preventing words being split up at the end of a line.

DATA May be placed anywhere in the program. Commas are used to separate

items, but are not necessary at the end of a line. Numeric and string data may be intermixed, but the program must call them in correct order. Strings do *not* need enclosing quotation marks unless they contain a comma, in which case quotation marks must be used.

RESTORE Sets the data pointer back to the first DATA statement.

END Not required in SWTP BASIC unless it is necessary to end the program in the middle of the listing. END may appear more than once and need not appear at all.

FOR . . . TO . . . NEXT . . . STEP The FOR and NEXT statements are used together to set up program loops. The variable in the FOR statement is set up for the number of times it is desired to pass through the loop, with a minimum of 1. When the NEXT is encountered, the variable is incremented by 1 if no STEP has been defined, and execution resumes at the statement following the FOR . . . TO. If the addition of the STEP develops a sum greater than the TO expression, execution passes to the line following the NEXT statement. STEP can be negative, in which case execution continues until the sum is *less* than the TO expression. FOR, NEXT and STEP may be expressions, but they are evaluated once only.

GOSUB After a subroutine, control passes to the line following the GOSUB statement.

GOTO An unqualified jump.

IF . . . THEN If the relation given in the IF is 'true', control is given to the line following the THEN. For example, IF X = 5 THEN 50 transfers control to line 50 if and only if X = 5. If the relation is false, control passes to the line following the IF — i.e. the rest of the line is not considered, even if it is a multiple-statement line.

It is also possible to execute a valid BASIC statement after the THEN. For instance, IF X = 5 PRINT "CORRECT". Note also that SWTP BASIC will assume the existence of THEN. Some demand that THEN be included.

INPUT Allows the user to enter data or strings. Various forms are allowed, as (e.g.) INPUT X, Y, Z which will prompt three times for numeric data, or INPUT Q$, R$ which will prompt twice for string input. Numerics entered for a string will be accepted and converted into a string.

PRINT and INPUT statements may be combined, as in: INPUT "GIVE ME A NUMBER", N. This prints out to the screen the message GIVE ME A NUMBER? (notice the addition of the '?' prompt), and waits for a numeric input. Another point to notice about this command is the comma following the message string; some BASICs have a semi-colon here.

LET This is optional.

ON . . . GOTO and **ON . . . GOSUB** These statements transfer control to the line numbers, depending on the value of the expression following ON. For example:

ON X + 1 GOTO 100, 200, 300, 400

If X = 0, control will transfer to line 100; if X = 1, to line 200; if X = 2, to 300 and if X = 3 to 400. Any other value for X will result in an error message.

REM Short for REMARK — an aid for human readers only. These lines may be omitted, as when BASIC is running, all lines beginning with REM are ignored.

Functions

ABS(X) returns the absolute value of X; i.e. can never be negative.

ATAN(X) returns the angle in radians whose tangent is X.

ASC(STRING) returns decimal value of first character in the string; e.g. ASC("?") gives 63.

CHR$(X) returns a single character equivalent to the decimal ASCII value of X. For example CHR$(65) gives A.

COS(X) returns cosine of X, which must be in radians.

DEF FNX(Y) = (exp) is where X can be any single letter, which names the function. It allows the creation of any function, e.g. DEF FNA(Y) = 3.14*Y ↑ 2. The Y is a dummy variable replaced by the actual variables desired when the function is called, e.g.

```
100 LET G = FNA(23)
```

will cause G to be assigned the value 1661.06 since Y was replaced by 23. If a variable occurs in the expression that is different from the dummy, then the current value of that variable is taken into account when evaluating.

EXP(X) returns the base of natural logarithms raised to the Xth power. The inverse of LOG(X).

INT(X) returns the greatest integer less than X, i.e. rounds down.

LEFT$(X$,N) returns a string of characters N characters long, commencing at the left-most in X$.

RIGHT$(X$,N) is as above, from the right-most.

MID$(X$,S,T) extracts from X$ a string of characters T characters long, starting at the Sth.

LEN(X$) returns the number of characters currently in X$, including spaces.

LOG(X) returns the natural logarithm of X.

RND & RND(0) return a random number between 0 and 1.

SGN(X) returns the sign of X; i.e., −4 produces −1; + 4 produces + 1.

SIN(X) returns the sine of the angle X (in radians).

SQR(X) returns the square root of X.

STR$(X) translates a numeric variable into a string.

VAL(X$) is the opposite: it returns as a numeric constant the first value encountered in X$.

TAB(X) moves the print position to the Xth place on the line. May be an expression. Note that if whatever is to be printed at a tab position is on another line, or following another PRINT statement, a semicolon (;) is needed following TAB in order to inhibit the L/F, C/R. For example, PRINT TAB(16);

TAN(X) returns the tangent of the angle X (in radians).

Note on graphics

No graphics have been included in these listings. This will no doubt cause considerable criticism in some quarters, but readers will appreciate that all graphics are machine-dependent. Including graphics would therefore restrict possible readers to the owners of one type of machine only, whereas omitting graphics allows these listings to be used by owners of all machines. To this end, wherever possible, machine-dependent commands have been omitted and the listings kept simple.

There is no doubt, however, that nearly all of the programs listed would be improved by graphics and/or some special commands. These must be left to the skill and requirements of individual owners. There is just no way that a single book can cater for all.

The few machine-dependent commands are restricted to the CHR$(X) variety, where the X executes non-printable functions at the terminal. A list of these follows; simply substitute with whatever executes the same function on your own machine. If you do not have a particular function available, merely omitting the command will generally have no effect on the program.

CHR$(5)	Cursor off
CHR$(6)	Erase to end of line
CHR$(7)	Bell
CHR$(8)	Cursor left
CHR$(9)	Cursor right
CHR$(10)	Cursor down
CHR$(11)	Cursor up
CHR$(12)	Change page
CHR$(13)	Carriage return
CHR$(14)	Scrolling mode
CHR$(15)	Not applicable
CHR$(16)	Home up
CHR$(17)	Tape reader on
CHR$(18)	Tape punch on
CHR$(19)	Tape reader off
CHR$(20)	Tape punch off
CHR$(21)	Cursor on
CHR$(22)	Erase to end of frame
CHR$(23)	Not applicable
CHR$(24)	Not applicable
CHR$(25)	Initialise terminal
CHR$(26)	Page mode
CHR$(27)	Cursor steady
CHR$(28)	Cursor blink
CHR$(29)	Reverse screen

Page/scroll

In page mode, when the cursor reaches the end of the bottom line, it continues printing at the beginning of the top line. Note that (at least on my machine) it does not automatically clear the screen; a separate command must be sent to do this,

which is a useful feature of one merely wishes to alter a few characters here and there. Similarly with the 'Home up' command, which merely flies the cursor to the top-left position without clearing the screen.

Paging is useful for automated semi-graphics; that is, for moving about printable characters. This is done with the program 'Slide' for instance. With the cursor switched off and the terminal in the page mode, the printable characters appear to slide about the screen. This is done by printing a string such as (e.g.) "A B C D E". If a carriage-return is now executed and the string "A BC D E" printed, the letter C will appear to move to the left — apparently without volition. This is the basis of many so-called graphics programs, including those run on some best-selling computers.

In scrolling mode, when the cursor reaches the end of the bottom line, the entire screen scrolls up one line, thus presenting a fresh clean line at the bottom upon which the cursor continues to print. The effect is like a continuous roll of paper issuing from a teletype.

Whether you like scrolling and use it a lot will depend to some extent upon the baud rate of your terminal. If it is very slow (say at 300 baud or less), the scrolling rate is just about at comfortable reading speed. If it is very fast (say at over 1200 baud), the screen fills in one or two seconds, after which the program pauses for you to read. In between those extremes, the scrolling effect can be very irritating indeed and you will not use scrolling very often.

Scrolling is used a lot in these programs because of the very high baud rate of my terminal. If you don't like it, it should be a simple matter to substitute page mode, or whatever takes your fancy — including graphics.

Sum-difference (0.5 K)

This, the simplest program in the book, will easily fit in any computer that supports BASIC, and is readily translatable into machine code for tiny computers that do not. It will even translate for programmable calculators, providing the text is omitted.

The game is of course suitable only for children. It is extremely simple, but unless you are in the secret, finding the two hidden numbers can be frustrating.

For budding programmers, it is interesting to observe that a PRINT statement can include computation, as in lines 110, 120 and 290. There is no need to do the calculation before printing out the answer, and the method shown can save program and variable space. Of course, if you want to do something else with the result of a calculation, it is better to save the answer rather than doing the calculation twice, but in this program it does not matter.

Notice also that the program has no exit, but will continue to loop forever, continuously posing problems until power is turned off, or until the user resets the machine. If you like, line 300 can be rewritten to ask the user if he wants to play again:

```
300  PRINT : PRINT "Type 1 to play again";
310  INPUT A
320  IF A = 1 THEN 10
330  END
```

If you have string facility and also use multiple statement lines, the above could be reduced to only one line:

```
300  PRINT : INPUT "Play again (Y-N)", A$:IF A$="Y" THEN 10
```

You will see lots of similar examples later in this book.

Lines 30 and 40 are interesting. Line 120 demands that X shall always be greater than Y, yet lines 10 and 20 do not guarantee this when they generate the two secret numbers. Therefore line 30 checks which is the greater and, if necessary, line 40 turns the two numbers around, putting the larger in X and the smaller in Y.

List of variables

X	first secret number	Y1	user's guess at second number
Y	second secret number	N	ongoing count of number of problems set
Z	temporary holding store	R	ongoing count of number of problems solved
X1	user's guess at first number		

Program listing

```
0010 X=INT(RND(0)*20)+1
0020 Y=INT(RND(0)*20)+1
0030 IF X>=Y THEN 50
0040 Z=X:X=Y:Y=Z
0050 N=N+1
0100 PRINT "I AM THINKING OF TWO NUMBERS BETWEEN 1 AND 20"
0110 PRINT "THEIR SUM IS ";X+Y
0120 PRINT "THEIR DIFFERENCE IS ";X-Y
0130 PRINT "WHAT ARE THE TWO NUMBERS?"
0140 INPUT X1:INPUT Y1
0150 IF X1<>X THEN 200
0160 IF Y1<>Y THEN 200
0170 PRINT "RIGHT!"
0180 R=R+1
0190 GOTO 290
0200 IF X1<>Y THEN 250
0210 IF Y1=X THEN 170
0250 PRINT "SORRY - YOU'RE WRONG."
0260 PRINT "THE NUMBERS WERE ";X;" AND ";Y
0290 PRINT "YOU ARE BATTING AT ";R*100/N;"%"
0300 PRINT :PRINT"LET'S PLAY AGAIN!":GOTO 10
```

A sample run

```
I AM THINKING OF TWO NUMBERS BETWEEN 1 AND 20
THEIR SUM IS 13
THEIR DIFFERENCE IS 5
WHAT ARE THE TWO NUMBERS?
8
5
SORRY - YOU'RE WRONG.
THE NUMBERS WERE 9  AND 4
YOU ARE BATTING AT 0 %

LET'S PLAY AGAIN!
I AM THINKING OF TWO NUMBERS BETWEEN 1 AND 20
THEIR SUM IS 10
THEIR DIFFERENCE IS 2
WHAT ARE THE TWO NUMBERS?
6
4
RIGHT!
YOU ARE BATTING AT 50 %

LET'S PLAY AGAIN!
I AM THINKING OF TWO NUMBERS BETWEEN 1 AND 20
THEIR SUM IS 20
THEIR DIFFERENCE IS 2
WHAT ARE THE TWO NUMBERS?
11
9
RIGHT!
YOU ARE BATTING AT 66.6666666 %

LET'S PLAY AGAIN!
I AM THINKING OF TWO NUMBERS BETWEEN 1 AND 20
THEIR SUM IS 10
THEIR DIFFERENCE IS 6
```

I am told what my mistake was!

can I get back to 100%?

Balancing act (1.2 K)

Although this is another nice simple little game suitable for any computer or programmable calculator, it is not all that easy to play well, and learners can be expected to make wild swings from side to side — in fact, just like the real situation. Children especially love this game.

If or when you get good at it, put a time-delay in the reaction loop:

```
210  W = W–D : D = P : NEXT M
```

The wobble (W) is then affected by the delayed reaction (D), and the current reaction (P) is stored for next time. This will make your reaction take effect always one move behind — that will take the starch out of anybody!

Line 130 is particularly interesting. The RND function returns a random number in the range 0 to .99999 and the effect of squaring this is to return most numbers in the lower end of the range, with occasional forays into the higher end. Multiplying by 8 and subtracting 4 means that the complete statement will now return a random number in the range −3 to + 3, thus causing a wobble of the cue to left or right.

List of variables

Q$ general input string
A number of moves aimed at
T tab position
M current move
W wobble left or right
P player's adjustment

Program listing

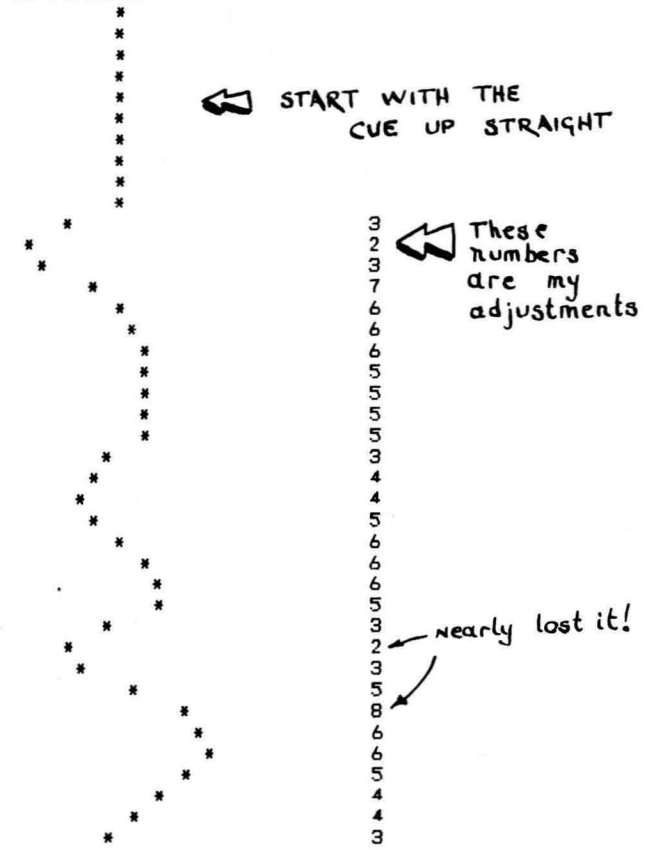
```
0010 PRINT TAB(10);"BALANCING ACT"
0020 PRINT TAB(10);"============"
0030 PRINT :PRINT
0040 INPUT "DO YOU WANT INSTRUCTIONS",Q$
0050 IF LEFT$(Q$,1)="Y" THEN GOSUB 9000
0060 PRINT "FOR HOW MANY MOVES DO YOU THINK YOU CAN BALANCE THE ";
0070 INPUT "BILLIARD CUE",A
0080 REM - BALANCE THE CUE
0090 T=10:FOR X=1 TO 10:PRINT TAB(T);"*":NEXT X
0100 REM - GAME LOOP ******************************
0110 FOR M=1 TO A
0120 IF W<0 THEN W=W-.5
0130 IF W=0 THEN X=RND:W=INT(X*X*8-4)
0140 IF W>0 THEN W=W+.5
0150 T=T+W:REM - INCREASE WOBBLE
0160 IF T<=0 THEN 500
0170 IF T>=20 THEN 500
0180 PRINT TAB(T);"*";TAB(30);
0190 INPUT P:P=P-5
0210 W=W-P:NEXT M
0220 PRINT :PRINT "CONGRATULATIONS - YOU DID IT!"
0230 PRINT :INPUT "PLAY AGAIN (Y-N)",Q$
0240 IF Q$="Y" THEN 60
0250 END
0500 PRINT :PRINT "BOO!  YOU DROPPED IT!":GOTO 230
9000 PRINT "IN THIS GAME YOU HAVE TO BALANCE A BILLIARD CUE ";
9010 PRINT "ON YOUR FOREHEAD.  IF IT STARTS TO WOBBLE TO THE ";
9020 PRINT "LEFT, THEN YOU OF COURSE MUST ALSO GO TO THE LEFT ";
9030 PRINT "SO AS TO GET YOURSELF UNDER THE CENTRE OF GRAVITY."
9040 PRINT "IF IT GOES RIGHT, YOU MUST GO RIGHT."
9050 PRINT "YOU USE THE NUMERICAL KEYS.  THE NUMBER 5 IS CENTRAL, ";
9060 PRINT "WITH THE NUMBERS 1-4 TO THE LEFT AND THE NUMBERS 6-9 ";
9070 PRINT "TO THE RIGHT."
9080 INPUT "     PRESS RETURN.....",Q$
9090 RETURN
```

MAKE IT FALL FASTER

TRY AND CORRECT IT

A sample run

```
DO YOU WANT INSTRUCTIONS? N
FOR HOW MANY MOVES DO YOU THINK YOU CAN BALANCE
THE BILLIARD CUE? 30
```




```
CONGRATULATIONS - YOU DID IT!

PLAY AGAIN? NO
READY
```

Pakistani pool (2 K)

As its name implies, this game originated in Pakistan. Instructions are included in the listing, but these can be omitted if desired, or if you can't quite fit the program into your machine. The same thing applies to the programs throughout this book — if you need to pare because your memory size is too small, take out the instructions first. (You can always keep them handy in a little booklet, or refer to this volume.)

If you still can't get a program to fit, take out all the REM statements (for REMARK), which are only in there to make the listing easier for you — the human — to follow. The computer doesn't need them! If you make a habit of this, you will have to be careful, however: make sure that another statement line somewhere doesn't cause a jump to the line that you take out. I have tried to avoid jumping to REM statements, but human nature being what it is, I could have slipped up.

Another way of saving memory is by multiple-statement lines, if your machine allows this. For example, lines 100 and 110 could go on one line, as could lines 120 and 130:

```
100  PRINT : PRINT "PLEASE TYPE YOUR NAMES" : P = 1
120  INPUT P$(P) : IF P$(P) = "" THEN 150
```

I have not done this throughout the listing, as it tends to make the print rather dense to read.

Line 40 is a handy trick. In response to line 30, users may type 'NO' or 'YES', or just N or Y. Line 40 looks only at the first letter of the input and acts accordingly, so even 'YES PLEASE' is acceptable. The program does not abort and user tension is avoided.

Line 130 avoids the necessity to input the number of players beforehand — names are simply typed in as long as you like and, when no more are to be entered, an extra tap on the RETURN key moves the program on.

List of variables

Q$	general response input
P	total number of players
P$(X)	names of individual players
C(X)	array containing each player's cash in hand

P(X)	array of each player's bet
X	general counting variable
C	total cash in the pool
H	number of heads thrown
T	number of tails thrown

Program listing

```
0010 PRINT "PAKISTANI POOL"
0020 PRINT "--------------"
0030 INPUT "DO YOU WANT INSTRUCTIONS",Q$
0040 IF LEFT$(Q$,1)="N" THEN 100
0050 PRINT "ON EACH ROUND, THE COMPUTER WILL SPIN 16 COINS."
0060 PRINT "THE PLAYERS BET ON THE NUMBER OF HEADS THAT THERE"
0070 PRINT "WILL BE. EACH PUTS $1 IN THE POOL AND THE PERSON"
0080 PRINT "GUESSING CORRECTLY SCOOPS THE POOL. IF NO-ONE"
0090 PRINT "GUESSES, THE CASH IN THE POOL GOES FORWARD TO THE "
0095 PRINT "NEXT ROUND."
0100 PRINT :PRINT "PLEASE TYPE YOUR NAMES"
0110 P=1
0120 INPUT P$(P)
0130 IF P$(P)="" THEN 150
0140 C(P)=20:P=P+1:GOTO 120
0150 PRINT "EACH PLAYER STARTS WITH $20"
0160 P=P-1
0170 FOR X=1 TO P
0180 PRINT P$(X);:INPUT " - YOUR GUESS",P(X)
0190 IF X=1 THEN 240
0200 FOR Y=1 TO X-1
0210 IF P(X)<>P(Y) THEN 230
0220 PRINT "THAT BET HAS BEEN TAKEN":GOTO 180
0230 NEXT Y
0240 C(X)=C(X)-1:C=C+1:NEXT X
0250 PRINT
0260 PRINT "HERE WE GO ...."
0270 PRINT :H=0:T=0
0280 FOR X=1 TO 16
0290 IF RND(0)>.5 THEN 310
0300 H=H+1:PRINT "H ";:GOTO 320
0310 T=T+1:PRINT "T ";
0320 NEXT X
0330 REM - CHECK BETS
0340 FOR X=1 TO P
0350 IF P(X)<>H THEN 400
0355 PRINT :PRINT
0360 PRINT P$(X);" WINS! HE/SHE NOW HAS ";
0370 C(X)=C(X)+C:C=0
0380 PRINT "$";C(X)
0390 GOTO 420
0400 NEXT X:PRINT
0410 PRINT H;"HEADS - NOBODY WINS. CASH IN POOL IS NOW $";C
0420 PRINT "CASH STATE -"
0430 FOR X=1 TO P
0440 PRINT P$(X);" - ";C(X)
0450 NEXT X
0460 FOR X=1 TO P
0470 IF C(X)>0 THEN 500
0480 PRINT "GAME ENDS. PLAYER(S) HAVE NO CASH"
0490 END
0500 NEXT X:GOTO 170
```