DOCUMENT RESUME

ED 268 959 IR 012 014

AUTHOR	Burns, Edward
TITLE	APPLE In-Service Programming for Teachers.
PUB DATE	[85]
NOTE	81p.
PUB TYPE	Guides - Non-Classroom Use (055) -- Computer Programs (101)
EDRS PRICE	MF01/PC04 Plus Postage.
DESCRIPTORS	Computer Assisted Instruction; *Computer Software; *Individualized Instruction; Inservice Teacher Education; *Programing; *Teacher Developed Materials
IDENTIFIERS	*Apple II; *BASIC Programing Language

ABSTRACT

This book is designed to provide teachers with techniques for entering and modifying BASIC programs on Apple computers. The underlying theme is that a teacher need not become a programmer to benefit from being able to use and modify BASIC programs. The key to the successful use of software in the classroom is the ability to individualize software programs and software content and the goal of this book is to demonstrate how BASIC programs can be entered and modified so as to individualize instruction. The programs provided are designed for use with the Apple II series. Specific instructions and short sample programs, each presented in building block fashion, are presented. Programing techniques are introduced via three kernel programs: (1) Math Tutorial; (2) Word Flash; and (3) Games and Graphics. (THC)

APPLE IN-SERVICE PROGRAMMING FOR TEACHERS

by

Edward Burns

University Center at Bingham.

PREFACE

The purpose of this book is to provide teachers with techniques for entering and modifying BASIC programs. The underlying theme of this book is that a teacher need not become a programmer to benefit from being able to use and modify BASIC programs.

The key to the successful use of software in the classroom is the ability to individualize software programs and, more importantly, software content. By far, the teacher is in the best position to determine what material is appropriate and how the computer can best be used to facilitate classroom learning and instruction.

The goal of this book is very straightforward: to demonstrate how BASIC programs can be entered and modified so as to individualize instruction. When one considers the tremendous number of free BASIC software programs available, learning a few basics of BASIC will unlock yet another door for the classroom teacher to improve classroom performance.

USING PROGRAMS

The programs in this book have been designed for use with the Apple II series (II, II+, IIe and IIc). Before you begin entering and modifying programs, do the following:

FIRST: Insert an initialized disk into the disk drive. If an initialized disk is not available, insert the DOS System Master disk that comes with the Apple.

If a disk or disk drive is not available, go to the next step.

SECOND: Switch on the Apple. If a disk or disk drive is not available, hold down the key labeled CONTROL while pressing the key labeled RESET. This will allow you to run programs in BASIC. Needless to say, without a disk or disk drive, you will not be able to save programs on disk.

With an initialized disk in the disk drive, the Apple switched on, and after a bit of whirring from the disk drive, you are ready to enter a BASIC program. A bracket, generated by the Apple, indicates that you are ready to enter a BASIC instruction:

```
]
```

Each program is presented in a building block fashion. Program instructions are added, one-by-one, to demonstrate the purpose of each instruction. Each program begins with a kernel program (i.e., 1.0, 2.0, 3.0). The following sample shows how a kernel program is entered and run:

0.0 Sample Kernel Program

A PRINT statement can be used to solve various arithmetic problems. The instruction is entered exactly as shown. The number before the instruction, 120, is the line number. Unless otherwise directed, the Apple begins processing the instruction with the lowest line number, then the next lowest, until all the instructions have been completed. After the line has been entered, press the RETURN key. This indicates that a single instruction will be sent to the Apple's memory.

```
]120 PRINT 9+6          (press RETURN key)
```

After an instruction or set of instructions has been entered, run the program by typing the word RUN. When you run the above single instruction program, the sum of 9+6 will appear:

```
]RUN
15
```

0.1 Using Variables

The first modification in the kernel program is the use of variables. This modification is made by simply re-entering line 120 and then adding lines 80 and 90. When these changes have been made, run the program and the sum of X and Y will be printed. Further modify the program by assigning different values to X and Y, and then run the program with these new values.

```
]80 X=9

]90 Y=6

]120 PRINT X+Y

]RUN
15
```

If you want to see your program on the screen, enter the command LIST and a listing of all your program instructions will appear.

```
]LIST

80 X = 9
90 Y = 6
120  PRINT X + Y
```

Remember: To add an instruction, modify an instruction or to correct an error, simply enter the line number and then the instruction. The instruction most recently entered will erase the previous instruction.

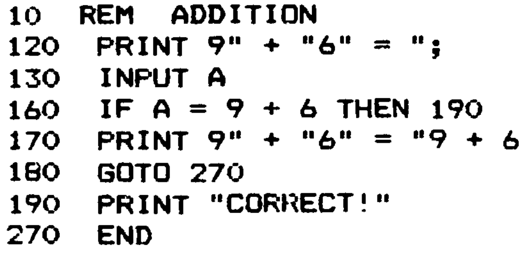
Now, before we actually begin, enter the word NEW and press the RETURN key. This will clear the Apple memory and thus prepare the Apple for the Math Tutorial program that follows.

```
]NEW
```

1.0 Math Tutorial

We will begin developing a math tutorial by first programming a single math problem. The problem consists of three parts: 1) the problem, 2) the response, and 3) feedback.

The below listing shows the interrelationship between the problem, response and feedback components. First, the problem 9 + 6 is printed. Second, the student enters an answer which is stored in a variable called A. Last, the response is evaluated. If the answer entered and stored in A is equal to 9 + 6, then CORRECT is printed. If the answer entered and stored in A is not equal to 9 + 6, the problem and correct answer are printed and the frame ends.



```
10  REM  ADDITION
120  PRINT 9" + "6" = ";
130  INPUT A
160  IF A = 9 + 6 THEN 190
170  PRINT 9" + "6" = "9 + 6
180  GOTO 270
190  PRINT "CORRECT!"
270  END
```

1.1 Entering Programs

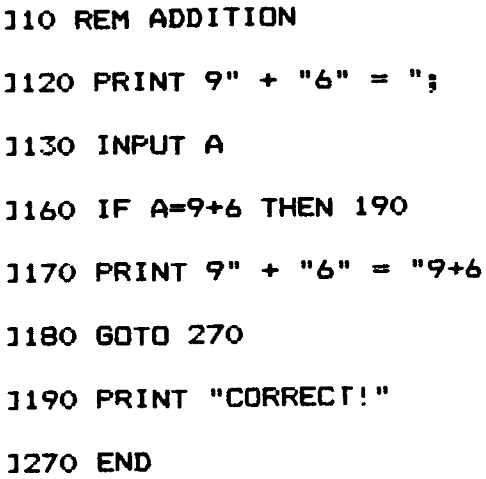
Every program is entered line-by-line. After an initialized disk has been placed in the disk drive and the Apple is turned on, a bracket will appear on the screen. The bracket indicates that the Apple is ready to accept a BASIC program.

The first line in the program is

10 REM ADDITION

Enter this line exactly as shown after the bracket. At the end of the line, press the RETURN key. This informs the Apple that one line or instruction has been completed. The RETURN key is pressed after each line or instruction is completed.

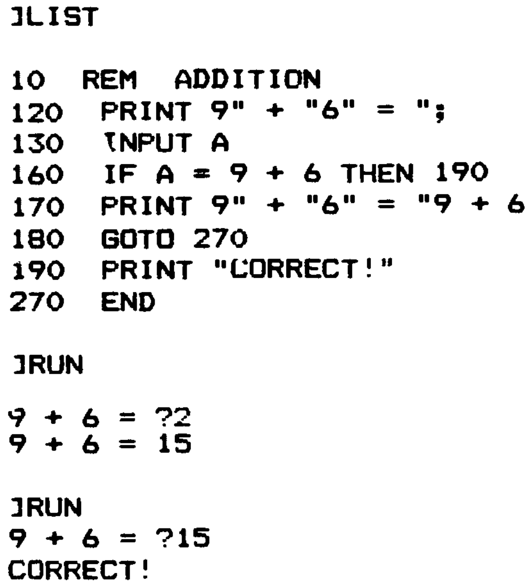
Enter the following lines exactly as shown. Don't forget to press the RETURN key at the end of each line.



```
]10 REM ADDITION
]120 PRINT 9" + "6" = ";
]130 INPUT A
]160 IF A=9+6 THEN 190
]170 PRINT 9" + "6" = "9+6
]180 GOTO 270
]190 PRINT "CORRECT!"
]270 END
```

1.2 Running Programs

After all the lines have been entered, type LIST. This command lists the entire program you have just entered on the screen. Now run the program by typing the command RUN. After an incorrect response, the problem and correct answer should be printed. Run the program again and enter the correct answer. If all is well, the prompt CORRECT will be printed.



```
]LIST

10  REM  ADDITION
120  PRINT 9" + "6" = ";
130  INPUT A
160  IF A = 9 + 6 THEN 190
170  PRINT 9" + "6" = "9 + 6
180  GOTO 270
190  PRINT "CORRECT!"
270  END

]RUN

9 + 6 = ?2
9 + 6 = 15

]RUN
9 + 6 = ?15
CORRECT!
```

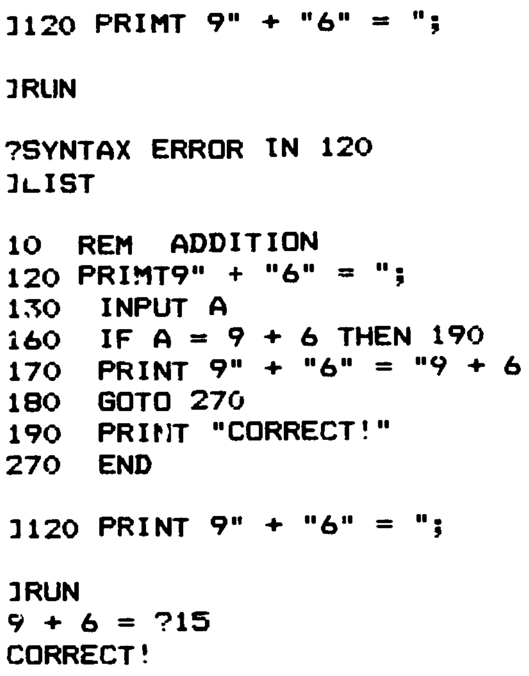
1.3 Bugs

Mistakes or bugs are easily taken care of by simply re-entering the guilty line. Re-type line 120 as follows:

```
120 PRIMT 9" + "6" = ";
```

Now, try and run the program. Because PRINT is spelled PRIMT, the Apple senses a syntax error and the program is interrupted. The program will not run correctly until this error is deleted.

The usual procedure for debugging is to first LIST the program, see where the error is, and then re-type the line containing the error.



```
]120 PRIMT 9" + "6" = ";

]RUN

?SYNTAX ERROR IN 120
]LIST

10  REM  ADDITION
120 PRIMT9" + "6" = ";
130  INPUT A
160  IF A = 9 + 6 THEN 190
170  PRINT 9" + "6" = "9 + 6
180  GOTO 270
190  PRINT "CORRECT!"
270  END

]120 PRINT 9" + "6" = ";

]RUN
9 + 6 = ?15
CORRECT!
```

1.4 Variables

We have already used a variable when we entered and stored an answer in the variable A. The Addition program can be given considerable more flexibility by using variables to print the problem. Add lines 80 and 90 as shown, and then modify lines 120, 160 and 170.

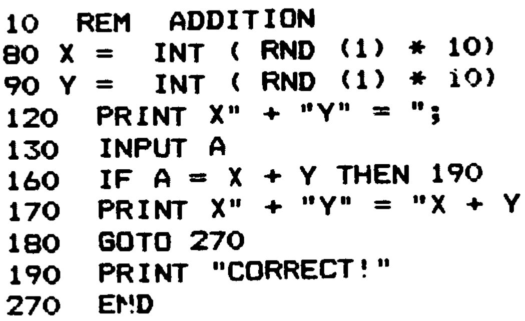
Now, with the variables X and Y, we can easily change the problem without re-entering lines 120, 160 and 170.

```
10  REM  ADDITION
80 X = 9
90 Y = 6
120  PRINT X" + "Y" = ";
130  INPUT A
160  IF A = X + Y THEN 190
170  PRINT X" + "Y" = "X + Y
180  GOTO 270
190  PRINT "CORRECT!"
270  END
```

1.5 Random Numbers

Rather than specifying the value of X and Y, we can leave that to the Apple by generating random numbers. Modify lines 80 and 90 exactly as shown. These modifications result in random numbers, ranging from 0 to 9, to be stored in X and Y.

A brief explanation: the RND(1) function results in a random number between 0 and .999999999. The number generated is multiplied by 10. As an example, RND(1) might result in .351326749. When multiplied by 10, the result is 3.51326749. Finally, the INT function changes this number to an integer. Thus, 3.51326749 becomes 3.



```
10  REM  ADDITION
80 X =  INT ( RND (1) * 10)
90 Y =  INT ( RND (1) * 10)
120  PRINT X" + "Y" = ";
130  INPUT A
160  IF A = X + Y THEN 190
170  PRINT X" + "Y" = "X + Y
180  GOTO 270
190  PRINT "CORRECT!"
270  END
```

1.6 Loops

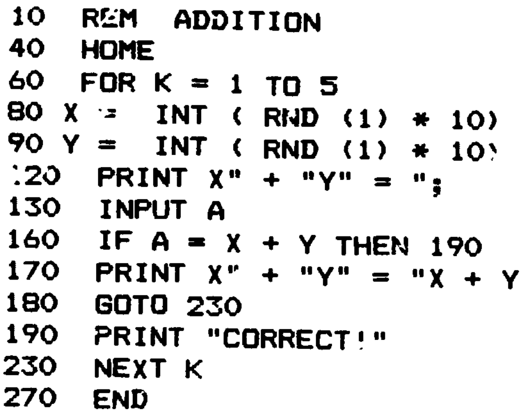
A loop in BASIC is a programming technique for repeating a series of instructions. We can use a loop to present a series of problems. As shown below, the loop begins in line 60 and ends in line 230. As indicated by the 5 in line 60, this loop results in the presentation of five problems. After the presentation of five problems, the loop ends.

```
10  REM  ADDITION
60  FOR K = 1 TO 5
80 X =  INT ( RND (1) * 10)
90 Y =  INT ( RND (1) * 10)
120  PRINT X" + "Y" = ";
130  INPUT A
160  IF A = X + Y THEN 190
170  PRINT X" + "Y" = "X + Y
180  GOTO 230
190  PRINT "CORRECT!"
230  NEXT K
270  END
```

1.7 Clearing the Screen

Each time the program is run, there is quite a bit of material on the screen. You might have a listing of the program, problems from previous runs, syntax errors, and who knows what. The HOME instruction is extremely useful in that it clears the screen and sends the cursor to the upper right-hand corner of the screen. You might call the HOME instruction the BASIC housecleaner.

Note that when the screen is cleared using HOME, the program is still in the Apple's memory. Although you can't see the program, it's there and ready to work.



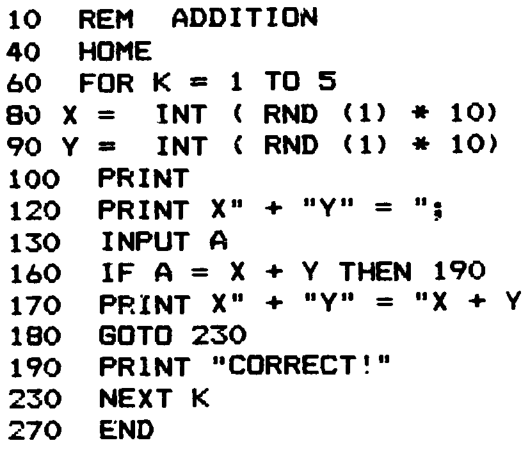
```
10  REM  ADDITION
40  HOME
60  FOR K = 1 TO 5
80 X =  INT ( RND (1) * 10)
90 Y =  INT ( RND (1) * 10)
120  PRINT X" + "Y" = ";
130  INPUT A
160  IF A = X + Y THEN 190
170  PRINT X" + "Y" = "X + Y
180  GOTO 230
190  PRINT "CORRECT!"
230  NEXT K
270  END
```

1.8 Skipping Lines

Sometimes the simplest of instructions can be the most useful. As the program now stands, the problems appear on the screen immediately after one another. We certainly don't want to cause undue confusion on the screen so it's a good idea to insert a blank line between problems. This is achieved by inserting a simple PRINT statement in line 100.

```
100 PRINT
```

A PRINT statement, followed by nothing, results in a line being skipped on the screen.

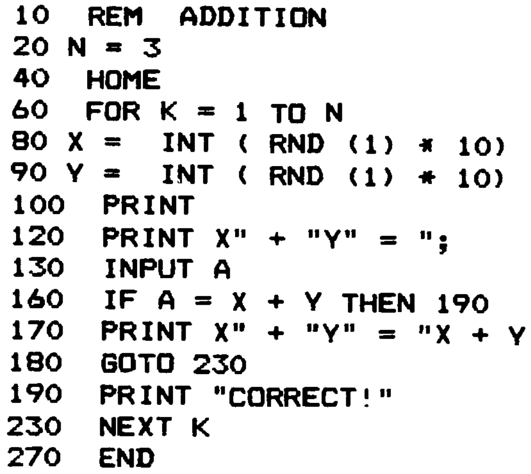


```
10  REM  ADDITION
40  HOME
60  FOR K = 1 TO 5
80 X =  INT ( RND (1) * 10)
90 Y =  INT ( RND (1) * 10)
100  PRINT
120  PRINT X" + "Y" = ";
130  INPUT A
160  IF A = X + Y THEN 190
170  PRINT X" + "Y" = "X + Y
180  GOTO 230
190  PRINT "CORRECT!"
230  NEXT K
270  END
```

1.10 Problem Number Control

We have used variables to store answers and to print addends. We can also use a variable to control the number of problems that are presented. As shown in the listing below, the variable N in line 20 determines the number of problems that are presented. In order to use N, line 60 must also be modified as shown.

Although N can be virtually any size (but be humane when presenting problems), N has been set to present only three problems in the below listing. Three problems is usually enough to determine whether the program is working after a change or modification.



```
10  REM  ADDITION
20 N = 3
40  HOME
60  FOR K = 1 TO N
80 X =  INT ( RND (1) * 10)
90 Y =  INT ( RND (1) * 10)
100  PRINT
120  PRINT X" + "Y" = ";
130  INPUT A
160  IF A = X + Y THEN 190
170  PRINT X" + "Y" = "X + Y
180  GOTO 230
190  PRINT "CORRECT!"
230  NEXT K
270  END
```

1.11 Addend Size

The size of each addend in problems is determined by lines 80 and 90. If addends ranging from 0 10 20 are desired, the following modifications are made:

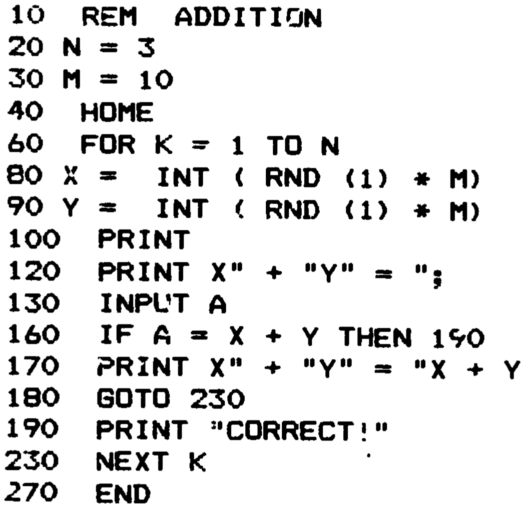
```
10  REM  ADDITION
20 N = 3
40  HOME
60  FOR K = 1 TO N
80 X =  INT ( RND (1) * 21)
90 Y =  INT ( RND (1) * 21)
100  PRINT
120  PRINT X" + "Y" = ";
130  INPUT A
160  IF A = X + Y THEN 190
170  PRINT X" + "Y" = "X + Y
180  GOTO 230
190  PRINT "CORRECT!"
230  NEXT K
270  END
```

If a smaller addend size is desired, say 0 to 4, then the following modifications are made:

```
10  REM  ADDITION
20 N = 3
40  HOME
60  FOR K = 1 TO N
80 X =  INT ( RND (1) * 5)
90 Y =  INT ( RND (1) * 5)
100  PRINT
120  PRINT X" + "Y" = ";
130  INPUT A
160  IF A = X + Y THEN 190
170  PRINT X" + "Y" = "X + Y
180  GOTO 230
190  PRINT "CORRECT!"
230  NEXT K
270  END
```

1.12 Variable Addend Size

Changing lines 80 and 90 each time a new maximum addend size is wanted can be tedious. This problem is solved by using a variable in lines 80 and 90. The variable M is used to determine the maximum size of each addend. The variable M is first defined (or set) in line 30. As shown, M is set to 10 which means that the addends randomly generated and sored in X and Y will range from 0 to 9.



```
10  REM  ADDITION
20 N = 3
30 M = 10
40  HOME
60  FOR K = 1 TO N
80 X =  INT ( RND (1) * M)
90 Y =  INT ( RND (1) * M)
100  PRINT
120  PRINT X" + "Y" = ";
130  INPUT A
160  IF A = X + Y THEN 190
170  PRINT X" + "Y" = "X + Y
180  GOTO 230
190  PRINT "CORRECT!"
230  NEXT K
270  END
```

1.13 More Screen Cleaning

We cleared the screen in line 40, but the problems then followed one after the other. We can clear the screen prior to the presentation of each problem by adding a HOME statement in line 70 (which is at the beginning of the loop that presents problems).

When HOME is inserted in line 70, what happens when the program is run and correct answers entered? See if you can explain what is happening to the CORRECT prompt before going on to the next section.

```
10  REM  ADDITION
20 N = 3
30 M = 10
40  HOME
60  FOR K = 1 TO N
70  HOME
80 X =  INT ( RND (1) * M)
90 Y =  INT ( RND (1) * M)
100  PRINT
120  PRINT X" + "Y" = ";
130  INPUT A
160  IF A = X + Y THEN 190
170  PRINT X" + "Y" = "X + Y
180  GOTO 230
190  PRINT "CORRECT!"
230  NEXT K
270  END
```

1.14 Delay Loops

When the screen is cleared before each problem, there is no delay after the prompt. Thus, after a correct is printed but it appears so briefly (before the screen is cleared for the next problem) that you can hardly see it. On a similar basis, following an incorrect response, the feedback appears so briefly that it is all but useless.

The problem is solved by adding a simple delay loop. A delay loop has one function: to give the Apple a busy task in order to cause a time delay. The delay loop in lines 210 and 220 causes the Apple to go from line 210, to 220, and back to 210 fifteen hundred times! A lot of work, but it only takes the Apple approximately two seconds.

Line 180 was changed from GOTO 230 to GOTO 210. Can you explain why? What happens if the program is run with the delay loop, line 180 is GOTO 210, and an incorrect problem is entered?

```
10  REM  ADDITION
20 N = 3
30 M = 10
40  HOME
60  FOR K = 1 TO N
70  HOME
80 X =  INT ( RND (1) * M)
90 Y =  INT ( RND (1) * M)
100  PRINT
120  PRINT X" + "Y" = ";
130  INPUT A
160  IF A = X + Y THEN 190
170  PRINT X" + "Y" = "X + Y
180  GOTO 210
190  PRINT "CORRECT!"
210  FOR D = 1 TO 1500
220  NEXT D
230  NEXT K
270  END
```

1.15 Summary Score

If we want to print the number correct after all the problems have been presented, we need to tally correct responses. This is accomplished by adding line 210. Each time a problem is correctly answered, 1 is added to variable C.

After all the problems have been presented, the screen is cleared in line 240. Finally, the number of problems presented (and stored in N), and the number correct are printed.

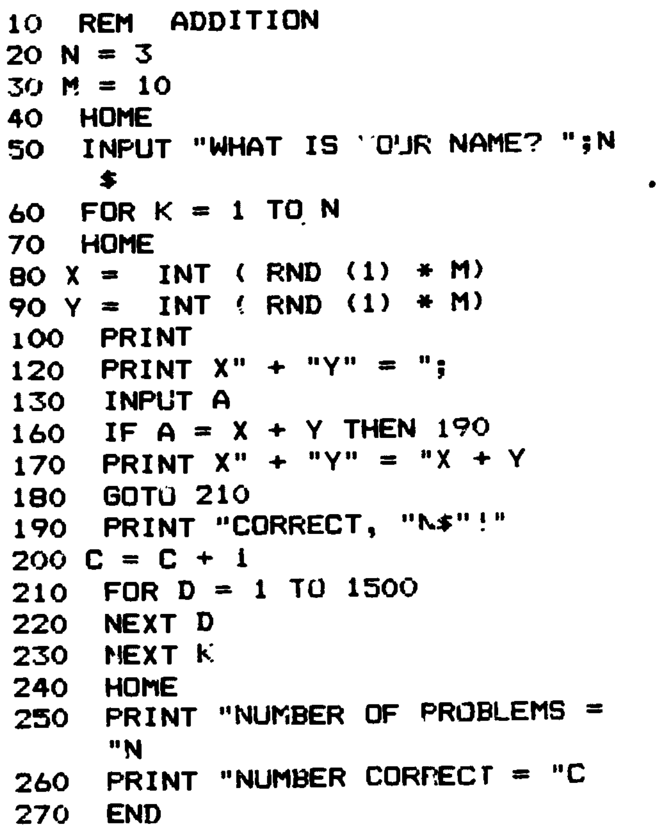
```
10  REM  ADDITION
20 N = 3
30 M = 10
40  HOME
60  FOR K = 1 TO N
70  HOME
80 X =  INT ( RND (1) * M)
90 Y =  INT ( RND (1) * M)
100  PRINT
120  PRINT X" + "Y" = ";
130  INPUT A
160  IF A = X + Y THEN 190
170  PRINT X" + "Y" = "X + Y
180  GOTO 210
190  PRINT "CORRECT!"
200 C = C + 1
210  FOR D = 1 TO 1500
220  NEXT D
230  NEXT K
240  HOME
250  PRINT "NUMBER OF PROBLEMS =
     "N
260  PRINT "NUMBER CORRECT = "C
270  END
```

1.16 Personal Individualization

The program can be individualized for each student by including the student's name when giving correct feedback. This is achieved by first inputting the student's name in line 50:

```
50 INPUT "WHAT IS YOUR NAME? ";N$
```

Next, change line 190 so that after the CORRECT prompt the student's name is printed. What is the purpose of the space after the comma in line 190?



```
10  REM  ADDITION
20 N = 3
30 M = 10
40  HOME
50  INPUT "WHAT IS YOUR NAME? ";N
    $
60  FOR K = 1 TO N
70  HOME
80 X =  INT ( RND (1) * M)
90 Y =  INT ( RND (1) * M)
100  PRINT
120  PRINT X" + "Y" = ";
130  INPUT A
160  IF A = X + Y THEN 190
170  PRINT X" + "Y" = "X + Y
180  GOTO 210
190  PRINT "CORRECT, "N$"!"
200 C = C + 1
210  FOR D = 1 TO 1500
220  NEXT D
230  NEXT K
240  HOME
250  PRINT "NUMBER OF PROBLEMS =
     "N
260  PRINT "NUMBER CORRECT = "C
270  END
```

1.17 INPUT Prompt (?) Suppression

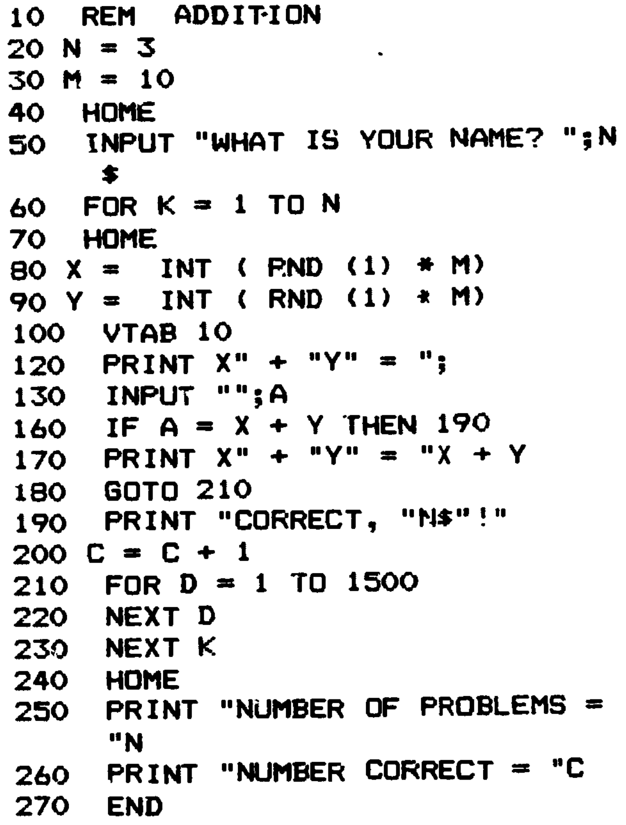
When an INPUT statement is used, the Apple prints a question mark prompt (?) to indicate that something should be entered. This prompt can be suppressed using a null string (a set of quotation marks), followed by a semicolon and then the variable in which a value is entered.

```
10  REM  ADDITION
20 N = 3
30 M = 10
40  HOME
50  INPUT "WHAT IS YOUR NAME? ";N
    $
60  FOR K = 1 TO N
70  HOME
80 X =  INT ( RND (1) * M)
90 Y =  INT ( RND (1) * M)
100  PRINT
120  PRINT X" + "Y" = ";
130  INPUT "";A
160  IF A = X + Y THEN 190
170  PRINT X" + "Y" = "X + Y
180  GOTO 210
190  PRINT "CORRECT, "N$"!"
200 C = C + 1
210  FOR D = 1 TO 1500
220  NEXT D
230  NEXT K
240  HOME
250  PRINT "NUMBER OF PROBLEMS =
     "N
260  PRINT "NUMBER CORRECT = "C
270  END
```

1.18 The VTAB Statement

The VTAB determines at which line the Apple will print. There are 24 lines on the Apple screen. If VTAB is set to 5 (VTAB 5), then the Apple begins printing at line 5. If VTAB is set to 10 (VTAB 10), then the Apple begins printing at line 10.

The PRINT statement in line 100 has been replaced with a VTAB statement which causes each problem to be printed at line 10. Because problems are printed at line 10, correct and incorrect feedback is given in the next line or line 11.



```
10  REM  ADDITION
20 N = 3
30 M = 10
40  HOME
50  INPUT "WHAT IS YOUR NAME? ";N
     $
60  FOR K = 1 TO N
70  HOME
80 X =  INT ( RND (1) * M)
90 Y =  INT ( RND (1) * M)
100  VTAB 10
120  PRINT X" + "Y" = ";
130  INPUT "";A
160  IF A = X + Y THEN 190
170  PRINT X" + "Y" = "X + Y
180  GOTO 210
190  PRINT "CORRECT, "N$"!"
200 C = C + 1
210  FOR D = 1 TO 1500
220  NEXT D
230  NEXT K
240  HOME
250  PRINT "NUMBER OF PROBLEMS =
     "N
260  PRINT "NUMBER CORRECT = "C
270  END
```

1.19 The HTAB Statement

The VTAB controls the line position and the HATB controls the column position. The Apple has 40 screen columns. If HTAB is set to 8 (HTAB *), printing begins in the eighth column. If VTAB is set to 15 (VTAB 15), then the Apple begins printing in the fifteenth column.

Two HTAB statements have been added to the program. First add line 110 and run the program. What happens when the feedback is printed. Now add line 150. Why has line 150 been added to the program?

```
10  REM  ADDITION
20 N = 3
30 M = 10
40  HOME
50  INPUT "WHAT IS YOUR NAME? ";N
    $
60  FOR K = 1 TO N
70  HOME
80 X =  INT ( RND (1) * M)
90 Y =  INT ( RND (1) * M)
100  VTAB 10
110  HTAB 15
120  PRINT X" + "Y" = ";
130  INPUT "";A
150  HTAB 15
160  IF A = X + Y THEN 190
170  PRINT X" + "Y" = "X + Y
180  GOTO 210
190  PRINT "CORRECT, "N$"!"
200 C = C + 1
210  FOR D = 1 TO 1500
220  NEXT D
230  NEXT K
240  HOME
250  PRINT "NUMBER OF PROBLEMS =
     "N
260  PRINT "NUMBER CORRECT = "C
270  END
```

1.20 Making those "little" but Important Changes

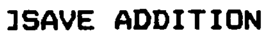
The greatest benefit of being able to use and modify your own software is that you can make whatever changes _you_ seem appropriate. In the Addition program, you might feel that it's a good idea to skip a line between the problem and feedback because it might make the output a little easier to read. This is accomplished easily enough by adding a PRINT statement in line 140.

```
10  REM  ADDITION
20 N = 3
30 M = 10
40  HOME
50  INPUT "WHAT IS YOUR NAME? ";N
     $
60  FOR K = 1 TO N
70  HOME
80 X =  INT ( RND (1) * M)
90 Y =  INT ( RND (1) * M)
100  VTAB 10
110  HTAB 15
120  PRINT X" + "Y" = ";
130  INPUT "";A
140  PRINT
150  HTAB 15
160  IF A = X + Y THEN 190
170  PRINT X" + "Y" = "X + Y
180  GOTO 210
190  PRINT "CORRECT, "N$"!"
200 C = C + 1
210  FOR D = 1 TO 1500
220  NEXT D
230  NEXT K
240  HOME
250  PRINT "NUMBER OF PROBLEMS =
     "N
260  PRINT "NUMBER CORRECT = "C
270  END
```

1.21 Saving Programs

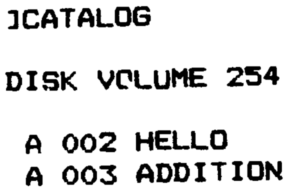
The program is completed and you are ready for a rest, but don't turn off the Apple or type NEW! If you do either of these, your program will be destroyed. Whenever you want to save a program on disk, use the SAVE command.

To use the SAVE command, type the word SAVE, followed by the name that you wish to give the program. As shown below, the program is saved on disk under the file name ADDITION. We could have used any name to save the program. Thus, if we had used SAVE EGOR, the program in the Apple's memory would have been saved under the file name EGOR.



```
]SAVE ADDITION
```

After a program is saved, it's probably a good idea to use the CATALOG command to print all the files on disk to be sure that the file you have attempted to save has actually been saved.



```
]CATALOG

DISK VOLUME 254

 A 002 HELLO
 A 003 ADDITION
```

1.22 Subtraction

The basic Addition program can be modified and individualized in a great many ways. The program is easily converted to a subtraction program by modifying lines 120, 160 and 170. In line 120 X+Y is the minuend and Y alone is the subtrahend so that the minuend is always equal to or greater than the subtrahend.

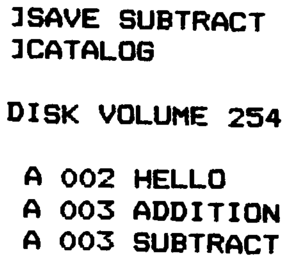
Although the REM (or remark statement) in line 10 is not prcessed, this line has also been changed so that when you see the listing you will quickly know that this is a subtraction tutorial.

```
10  REM  SUBTRACTION
20 N = 3
30 M = 10
40  HOME
50  INPUT "WHAT IS YOUR NAME? ";N
    $
60  FOR K = 1 TO N
70  HOME
80 X =  INT ( RND (1) * M)
90 Y =  INT ( RND (1) * M)
100  VTAB 10
110  HTAB 15
120  PRINT X + Y" - "Y" = ";
130  INPUT "";A
140  PRINT
150  HTAB 15
160  IF A = X THEN 190
170  PRINT X + Y" - "Y" = "X
180  GOTO 210
190  PRINT "CORRECT, "N$"!"
200 C = C + 1
210  FOR D = 1 TO 1500
220  NEXT D
230  NEXT K
240  HOME
250  PRINT "NUMBER OF PROBLEMS =
     "N
260  PRINT "NUMBER CORRECT = "C
270  END
```

1.23 Disk Business

Before you go on to the next program modification, you might want to save the Subtraction program on disk. As a matter of fact, each time you create a relatively unique program (e.g., subtraction, multiplication, division game, etc.), it is an execllent idea to save the program on disk. This is especially important if you clear the Apple's memory in order to enter (or LOAD) a new program.

As with the subtraction program, the SAVE command is used to save the program on disk. After the program has been saved, give the CATALOG command to be sure that the program is on disk.



```
]SAVE SUBTRACT
]CATALOG

DISK VOLUME 254

 A 002 HELLO
 A 003 ADDITION
 A 003 SUBTRACT
```

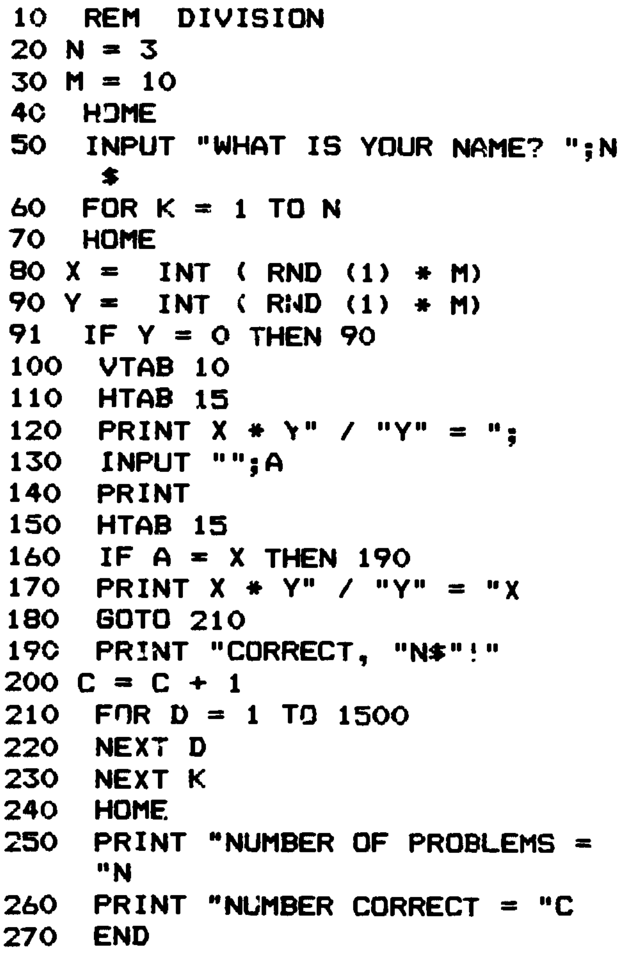
1.24 Multiplication

The BASIC operator for multiplication is an asterisk or *. Lines 120, 160 and 170 are changed to present a multiplication tutorial. In line 170, why is an X printed on the screen to show multiplication, but an * is used to actually multiply X and Y?

```
10  REM  MULTIPLICATION
20 N = 3
30 M = 10
40  HOME
50  INPUT "WHAT IS YOUR NAME? ";N
    $
60  FOR K = 1 TO N
70  HOME
80 X =  INT ( RND (1) * M)
90 Y =  INT ( RND (1) * M)
100  VTAB 10
110  HTAB 15
120  PRINT X" X "Y" = ";
130  INPUT "";A
140  PRINT
150  HTAB 15
160  IF A = X * Y THEN 190
170  PRINT X" X "Y" = "X * Y
180  GOTO 210
190  PRINT "CORRECT, "N$"!"
200 C = C + 1
210  FOR D = 1 TO 1500
220  NEXT D
230  NEXT K
240  HOME
250  PRINT "NUMBER OF PROBLEMS =
     "N
260  PRINT "NUMBER CORRECT = "C
270  END
```

1.25 Division

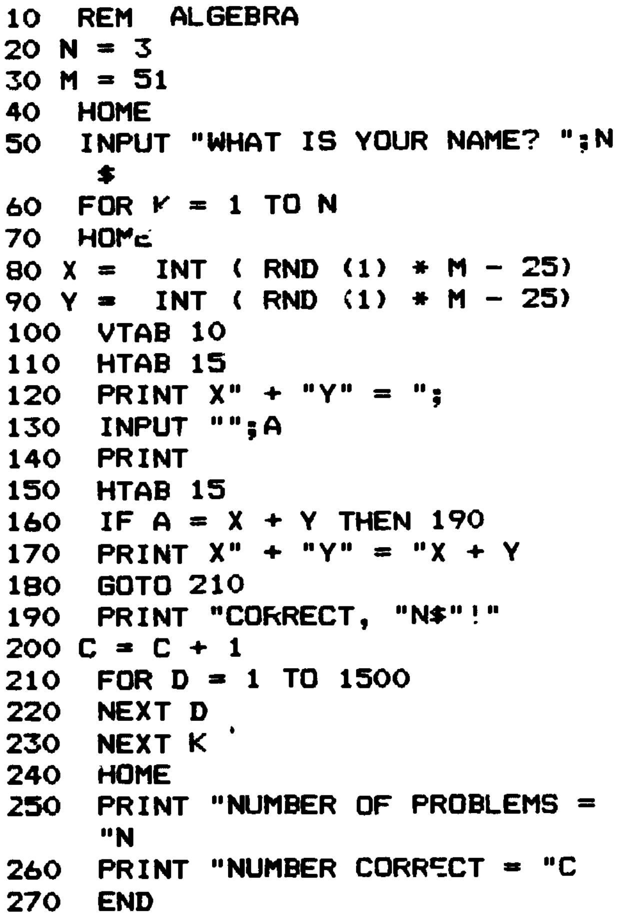
The slash symbol indicates division in BASIC. Thus, 10 / 2 results in a quotient of 5. Lines 120, 160 and 170 are modified in order to present a division tutorial. Line 91 is added to insure that the divisor (the variable Y) is never 0. For each problem, the dividend is the product of X and Y, and the quotient is X.



```
10  REM  DIVISION
20 N = 3
30 M = 10
40  HOME
50  INPUT "WHAT IS YOUR NAME? ";N
    $
60  FOR K = 1 TO N
70  HOME
80 X =  INT ( RND (1) * M)
90 Y =  INT ( RND (1) * M)
91  IF Y = 0 THEN 90
100  VTAB 10
110  HTAB 15
120  PRINT X * Y" / "Y" = ";
130  INPUT "";A
140  PRINT
150  HTAB 15
160  IF A = X THEN 190
170  PRINT X * Y" / "Y" = "X
180  GOTO 210
190  PRINT "CORRECT, "N$"!"
200 C = C + 1
210  FOR D = 1 TO 1500
220  NEXT D
230  NEXT K
240  HOME
250  PRINT "NUMBER OF PROBLEMS =
     "N
260  PRINT "NUMBER CORRECT = "C
270  END
```

1.26 Algebra

When the program is modified as shown below, M is set to 51 and the random numbers generated and stored in X and Y range from -25 to 25. First change lines 30, 80 and 90 as shown. Then modify lines 120, 160 and 170 to present addition problems. The result will be problem involving the addition of positive and negative integers, such that each addend is an integer ranging from -25 to 25.



```
10  REM  ALGEBRA
20 N = 3
30 M = 51
40  HOME
50  INPUT "WHAT IS YOUR NAME? ";N
     $
60  FOR K = 1 TO N
70  HOME
80 X =  INT ( RND (1) * M - 25)
90 Y =  INT ( RND (1) * M - 25)
100  VTAB 10
110  HTAB 15
120  PRINT X" + "Y" = ";
130  INPUT "";A
140  PRINT
150  HTAB 15
160  IF A = X + Y THEN 190
170  PRINT X" + "Y" = "X + Y
180  GOTO 210
190  PRINT "CORRECT, "N$"!"
200 C = C + 1
210  FOR D = 1 TO 1500
220  NEXT D
230  NEXT K
240  HOME
250  PRINT "NUMBER OF PROBLEMS =
     "N
260  PRINT "NUMBER CORRECT = "C
270  END
```

1.27 Algebra: Addition and Subtraction

The key to presenting a combination of addition and subtraction problems using negative and positive integers is the addition of variable P in line 91. This variable determines which type of problem is printed (lines 111 to 122), and how the problem is evaluated (lines 151 to 183).

```
10  REM  ALGEBRA
20 N = 3
30 M = 51
40  HOME
50  INPUT "WHAT IS YOUR NAME? ";N
    $
60  FOR K = 1 TO N
70  HOME
80 X =  INT ( RND (1) * M - 25)
90 Y =  INT ( RND (1) * M - 25)
91 P =  RND (1)
100  VTAB 10
110  HTAB 15
111  IF P > .5 THEN 122
120  PRINT X" + "Y" = ";
121  GOTO 130
122  PRINT X" - "Y" = ";
130  INPUT "";A
140  PRINT
150  HTAB 15
151  IF P > .5 THEN 181
160  IF A = X + Y THEN 190
170  PRINT X" + "Y" = "X + Y
180  GOTO 210
181  IF A = X - Y THEN 190
182  PRINT X" - "Y" = "X - Y
183  GOTO 210
190  PRINT "CORRECT, "N$"!"
200 C = C + 1
210  FOR D = 1 TO 1500
220  NEXT D
230  NEXT K
240  HOME
250  PRINT "NUMBER OF PROBLEMS =
     "N
260  PRINT "NUMBER CORRECT = "C
270  END
```

2.0 Word Flash

The Word Flash program is actually a tachistoscope that presents words on the screen for a specified period of time. After a word is presented, the screen is cleared and the student must enter the exact word that was presented. The Word Flash program is easily modified to accommodate words from specific content areas (e.g., science, math, social studies, etc.).

The first step in developing our Word Flash program is to first create a frame that presents a word (or sting) on the screen, evaluates a word entered by the student, and then provides feedback.

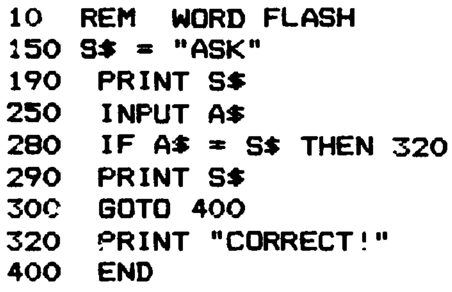
In the frame shown below, the word ASK is first shown on the screen. The student must then enter the word ASK and then press the RETURN key. If the word entered and stored in the string variable A$ is equal to the string ASK, CORRECT is printed.

```
10  REM  WORD FLASH
190  PRINT "ASK"
250  INPUT A$
280  IF A$ = "ASK" THEN 320
290  PRINT "ASK"
300  GOTO 400
320  PRINT "CORRECT!"
400  END
```

2.1 String Variables

A string variable is indicated by a dollar sign. The following are string variables: A$, S$, S$(1), W$. A string variable holds a "string" of characters such as words. In the Word Flash program string variables are used to store the words flashed on the screen and the student's response.

The word to be stored in the string varibale S$ is determined in line 150. The string "ASK" is then replaced by the string S$ in lines 190, 280 and 290. Rather than changing these lines each time a new word is presented, we need only change the word to be stored in S$ in line 150.



```
10  REM  WORD FLASH
150 S$ = "ASK"
190  PRINT S$
250  INPUT A$
280  IF A$ = S$ THEN 320
290  PRINT S$
300  GOTO 400
320  PRINT "CORRECT!"
400  END
```

2.2 READ and DATA Statements

Words can be read into the S$ variable from DATA statements by means of READ statements. The READ statement in line 150 causes the word ASK, which is contained in the DATA statement in line 410, to be read and stored in S$.

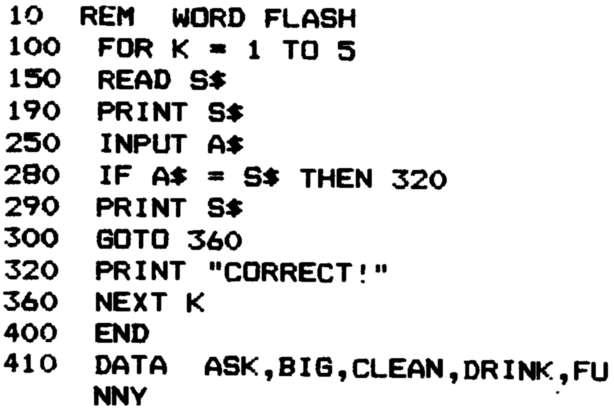
```
10  REM  WORD FLASH
150  READ S$
190  PRINT S$
250  INPUT A$
280  IF A$ = S$ THEN 320
290  PRINT S$
300  GOTO 400
320  PRINT "CORRECT!"
400  END
410  DATA  ASK
```

2.3 Loops

A loop can be used to present a series of problems. The loop below begins in line 100 and ends in line 360. As indicated by the 5 in line 100, this loop presents problems.

Because five problems are given, five words must be read from DATA statements. For every item specified in a READ statement there must be a corresponding item in a DATA statement. What happens when the program is run but only the word ASK is included in the DATA statement?

Because we don want the program to end after an incorrect response, line 300 must also be changed.



```
10  REM  WORD FLASH
100  FOR K = 1 TO 5
150  READ S$
190  PRINT S$
250  INPUT A$
280  IF A$ = S$ THEN 320
290  PRINT S$
300  GOTO 360
320  PRINT "CORRECT!"
360  NEXT K
400  END
410  DATA  ASK,BIG,CLEAN,DRINK,FU
     NNY
```

2.4 Clearing the Screen

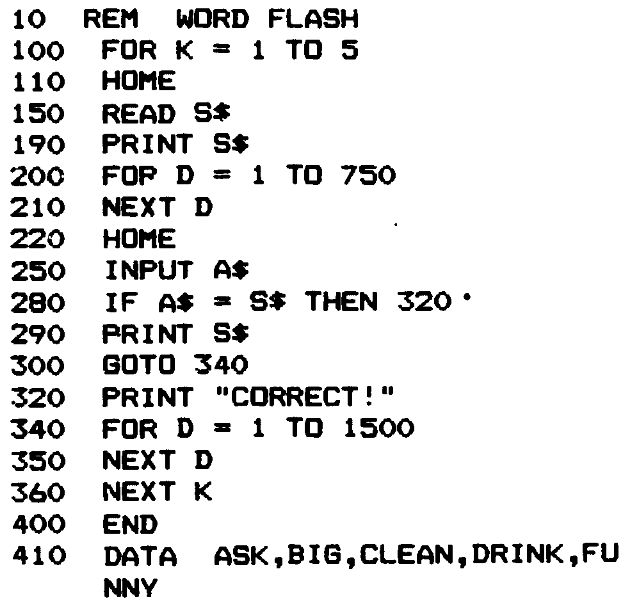
In order to present each word on a clear screen, the HOME command is added in line 110. Before making any additional changes, run the program. What happens after a correct or incorrect response?

The delay loop is needed in lines 340 and 350 so that the student has sufficient time to read the feedback. Also, the GOTO statement in line 300 must be changed in order to direct control to the delay loop following an incorrect response.

```
10   REM  WORD FLASH
100  FOR K = 1 TO 5
110  HOME
150  READ S$
190  PRINT S$
250  INPUT A$
280  IF A$ = S$ THEN 320
290  PRINT S$
300  GOTO 340
320  PRINT "CORRECT!"
340  FOR D = 1 TO 1500
350  NEXT D
360  NEXT K
400  END
410  DATA  ASK,BIG,CLEAN,DRINK,FU
     NNY
```

2.5 Flash Control

Lines 200, 210 and 220 determines the length of time each word is flashed on the screen. After a word is printed in line 190, the delay loop in lines 200 and 210 causes a delay of approximately one second. Immediately following this delay, the screen is cleared by means of the HOME statement in line 220.

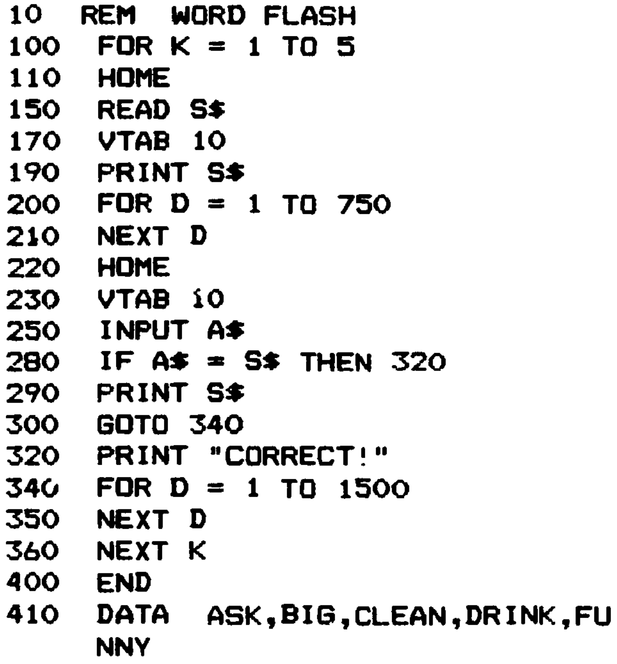


```
10  REM  WORD FLASH
100  FOR K = 1 TO 5
110  HOME
150  READ S$
190  PRINT S$
200  FOR D = 1 TO 750
210  NEXT D
220  HOME
250  INPUT A$
280  IF A$ = S$ THEN 320
290  PRINT S$
300  GOTO 340
320  PRINT "CORRECT!"
340  FOR D = 1 TO 1500
350  NEXT D
360  NEXT K
400  END
410  DATA  ASK,BIG,CLEAN,DRINK,FU
     NNY
```

2.6 Row Screen Position

The VTAB statement is used to position the cursor in the tenth row for each problem. As mentioned before, there are 24 rows on the Apple's screen.

The VTAB is used before the word is printed in line 190. The VTAB must also be used after the HOME statement in line 220. What happens if the VTAB is not used in line 230?

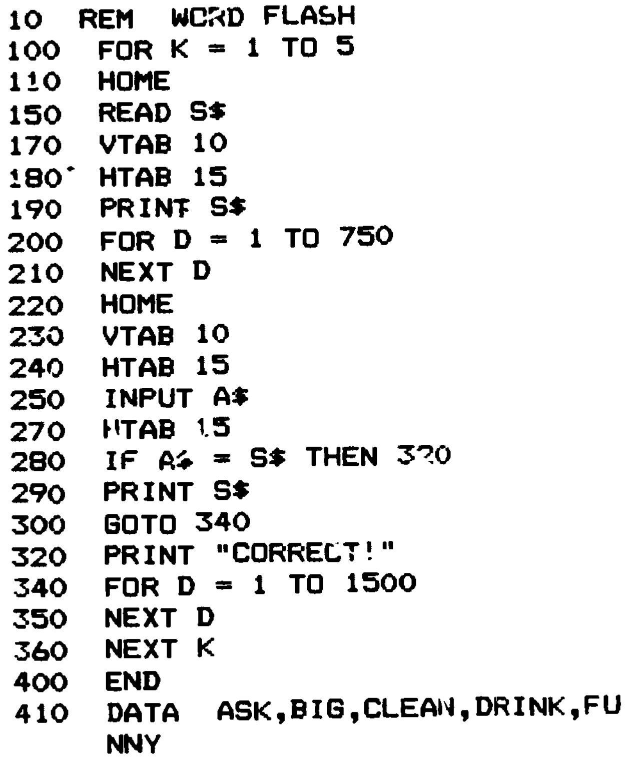


```
10  REM  WORD FLASH
100  FOR K = 1 TO 5
110  HOME
150  READ S$
170  VTAB 10
190  PRINT S$
200  FOR D = 1 TO 750
210  NEXT D
220  HOME
230  VTAB 10
250  INPUT A$
280  IF A$ = S$ THEN 320
290  PRINT S$
300  GOTO 340
320  PRINT "CORRECT!"
340  FOR D = 1 TO 1500
350  NEXT D
360  NEXT K
400  END
410  DATA  ASK,BIG,CLEAN,DRINK,FU
     NNY
```

2.7 Column Screen Position

The Apple screen is ccmprised on 24 rows and 40 columns. The HTAB designates column position. Lines 180, 240 and 270 sets the cursor to print in the fifteenth row.

Because each HTAB only determines the column position for the next item printed, HTAB's must be used before the stimulus word is printed, before the student's response is entered, and before feedback is given.



```
10  REM  WORD FLASH
100  FOR K = 1 TO 5
110  HOME
150  READ S$
170  VTAB 10
180  HTAB 15
190  PRINT S$
200  FOR D = 1 TO 750
210  NEXT D
220  HOME
230  VTAB 10
240  HTAB 15
250  INPUT A$
270  HTAB 15
280  IF A$ = S$ THEN 320
290  PRINT S$
300  GOTO 340
320  PRINT "CORRECT!"
340  FOR D = 1 TO 1500
350  NEXT D
360  NEXT K
400  END
410  DATA  ASK,BIG,CLEAN,DRINK,FU
     NNY
```

2.8 Array Variables

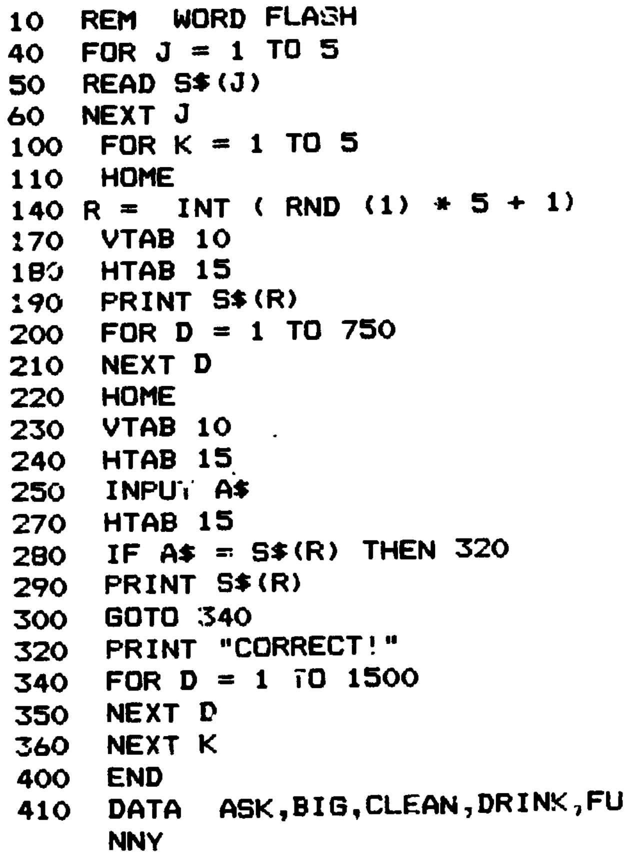
Rather than reading each word into the S$ variable each time through the loop, the words can be read into an array prior to presenting the problem. As a result of the loop in lines 40, 50 and 60, ASK is stored in S$(1), BIG in S$(2), CLEAN in S$(3), DRINK in S$(4), and FUNNY in S$(5).

Lines 190, 280 and 290 must be changed to include the array variable S$(K). The K variable is incremented each time through the loop beginning in line 100 (1, 2, 3, 4 and 5) so that the five words are presented in sequential order.

```
10  REM  WORD FLASH
40  FOR J = 1 TO 5
50  READ S$(J)
60  NEXT J
100  FOR K = 1 TO 5
110  HOME
170  VTAB 10
180  HTAB 15
190  PRINT S$(K)
200  FOR D = 1 TO 750
210  NEXT D
220  HOME
230  VTAB 10
240  HTAB 15
250  INPUT A$
270  HTAB 15
280  IF A$ = S$(K) THEN 320
290  PRINT S$(K)
300  GOTO 340
320  PRINT "CORRECT!"
340  FOR D = 1 TO 1500
350  NEXT D
360  NEXT K
400  END
410  DATA  ASK,BIG,CLEAN,DRINK,FU
     NNY
```

2.9 Random Word Selection

The statement in line 140 generates a random number ranging from 1 to 5. If 2 is generated, then the word in S$(2) is printed (see line 190. If the random number generated and stored in R is 4, then S$(R) is printed. Because R is 4, and the word DRINK is stored in S$(4), the word DRINK is printed in line 190.



```
10  REM  WORD FLASH
40  FOR J = 1 TO 5
50  READ S$(J)
60  NEXT J
100  FOR K = 1 TO 5
110  HOME
140 R =  INT ( RND (1) * 5 + 1)
170  VTAB 10
180  HTAB 15
190  PRINT S$(R)
200  FOR D = 1 TO 750
210  NEXT D
220  HOME
230  VTAB 10
240  HTAB 15
250  INPUT A$
270  HTAB 15
280  IF A$ = S$(R) THEN 320
290  PRINT S$(R)
300  GOTO 340
320  PRINT "CORRECT!"
340  FOR D = 1 TO 1500
350  NEXT D
360  NEXT K
400  END
410  DATA  ASK,BIG,CLEAN,DRINK,FU
     NNY
```

2.10 Word Number

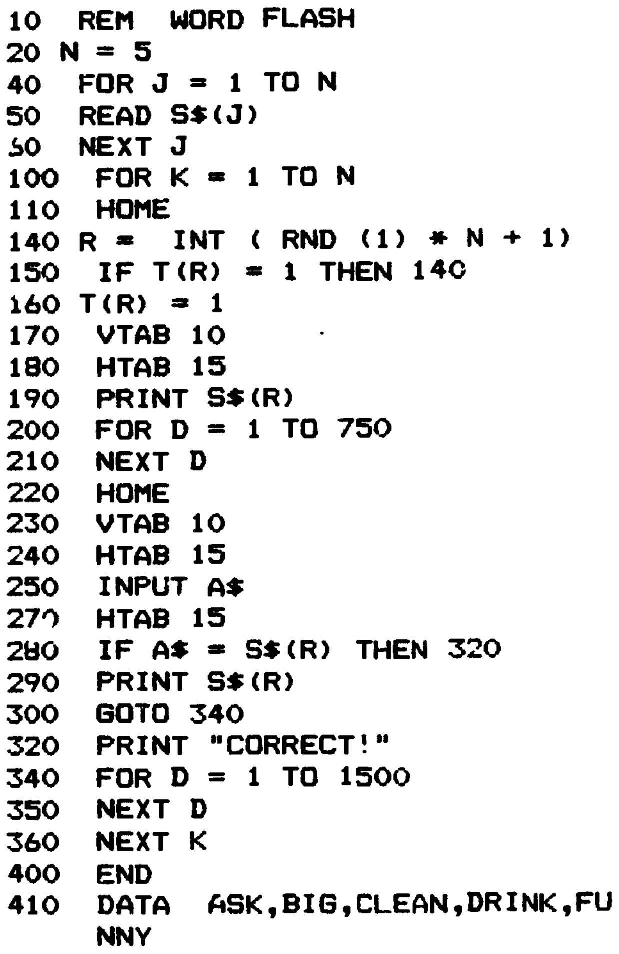
The number of words read and presented is more easily controlled by using a varibale to determine the number of words. The variable N in line 20 indicates how many words are to be read and presented.

The variable N is added to line 40 to control the number of words read into the S$(J) array; to control the number of problem presented in line 100; and to control the range of the random number generated and stored in R in line 140.

```
10  REM  WORD FLASH
20 N = 5
40  FOR J = 1 TO N
50  READ S$(J)
60  NEXT J
100  FOR K = 1 TO N
110  HOME
140 R =  INT ( RND (1) * N + 1)
170  VTAB 10
180  HTAB 15
190  PRINT S$(R)
200  FOR D = 1 TO 750
210  NEXT D
220  HOME
230  VTAB 10
240  HTAB 15
250  INPUT A$
270  HTAB 15
280  IF A$ = S$(R) THEN 320
290  PRINT S$(R)
300  GOTO 340
320  PRINT "CORRECT!"
340  FOR D = 1 TO 1500
350  NEXT D
360  NEXT K
400  END
410  DATA  ASK,BIG,CLEAN,DRINK,FU
     NNY
```

2.11 Unique Word Sequence

The array in lines 150 and 160 causes each word to be presented only once. As soon as a random number is generated and stored in R, T(R) is evaluated. If T(R) is equal to 1 (which indicates that the word designated by S$(R) has already been used, a new number is generated. If R has not been used before, a 1 is added to T(R) and the word in S$(R) is printed.



```
10  REM  WORD FLASH
20 N = 5
40  FOR J = 1 TO N
50  READ S$(J)
60  NEXT J
100  FOR K = 1 TO N
110  HOME
140 R =  INT ( RND (1) * N + 1)
150  IF T(R) = 1 THEN 140
160 T(R) = 1
170  VTAB 10
180  HTAB 15
190  PRINT S$(R)
200  FOR D = 1 TO 750
210  NEXT D
220  HOME
230  VTAB 10
240  HTAB 15
250  INPUT A$
270  HTAB 15
280  IF A$ = S$(R) THEN 320
290  PRINT S$(R)
300  GOTO 340
320  PRINT "CORRECT!"
340  FOR D = 1 TO 1500
350  NEXT D
360  NEXT K
400  END
410  DATA  ASK,BIG,CLEAN,DRINK,FU
     NNY
```

2.12 Name Individualization

As with the Math Tutorial, the program can be individualized by entering the student's name. Lines 70 and 90 are added, and the statement in line 320 is modified to include the student's name.

```
10  REM  WORD FLASH
20 N = 5
40  FOR J = 1 TO N
50  READ S$(J)
60  NEXT J
70  HOME
90  INPUT "WHAT IS YOUR NAME? ";N
     $
100  FOR K = 1 TO N
110  HOME
140 R =  INT ( RND (1) * N + 1)
150  IF T(R) = 1 THEN 140
160 T(R) = 1
170  VTAB 10
180  HTAB 15
190  PPINT S$(R)
200  FOR D = 1 TO 750
210  NEXT D
220  HOME
230  VTAB 10
240  HTAB 15
250  INPUT A$
270  HTAB 15
280  IF A$ = S$(R) THEN 320
290  PRINT S$(R)
300  GOTO 340
320  PRINT "CORRECT, "N$"!"
340  FOR D = 1 TO 1500
350  NEXT D
360  NEXT K
400  END
410  DATA  ASK,BIG,CLEAN,DRINK,FU
     NNY
```

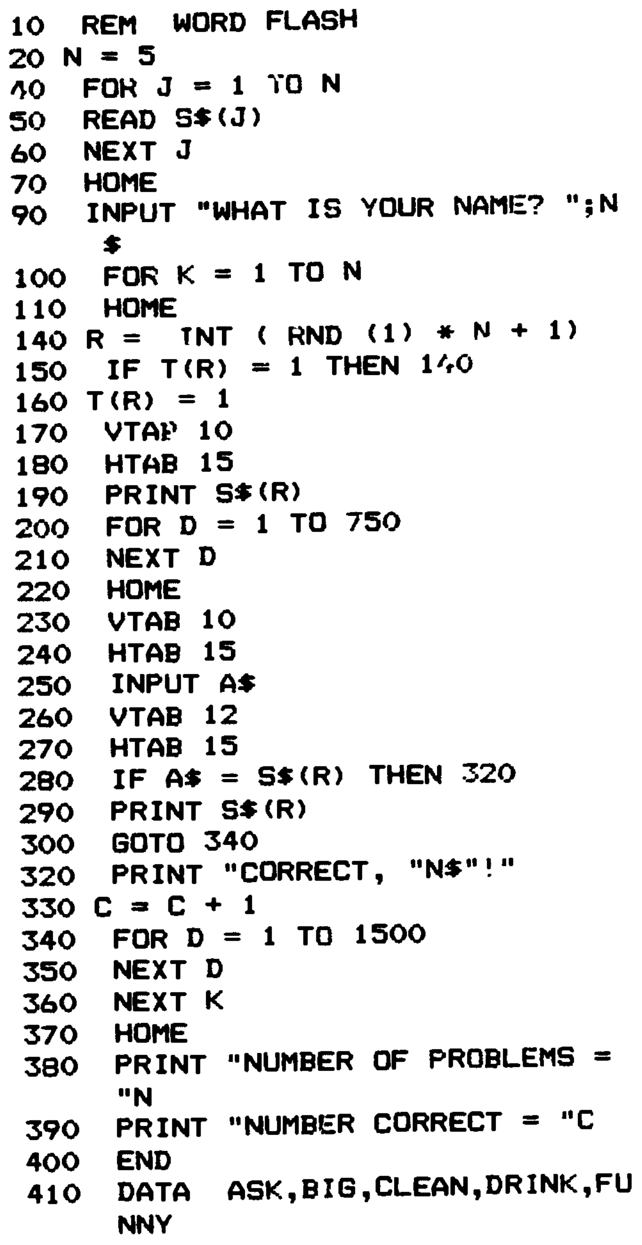
2.13 Spacing

After a correct or incorrect response, the feedback is printed immediately below the response entered. The feedback is much easier to read if it is set apart from the response entered by a blank line. This is accomplished by printing the feedback in the twelfth line as indicated in line 260.

```
10  REM  WORD FLASH
20 N = 5
40  FOR J = 1 TO N
50  READ S$(J)
60  NEXT J
70  HOME
90  INPUT "WHAT IS YOUR NAME? ";N
     $
100  FOR K = 1 TO N
110  HOME
140 R =  INT ( RND (1) * N + 1)
150  IF T(R) = 1 THEN 140
160 T(R) = 1
170  VTAB 10
180  HTAB 15
190  PRINT S$(R)
200  FOR D = 1 TO 750
210  NEXT D
220  HOME
230  VTAB 10
240  HTAB 15
250  INPUT A$
260  VTAB 12
270  HTAB 15
280  IF A$ = S$(R) THEN 320
290  PRINT S$(R)
300  GOTO 340
320  PRINT "CORRECT, "N$"!"
340  FOR D = 1 TO 1500
350  NEXT D
360  NEXT K
400  END
410  DATA  ASK,BIG,CLEAN,DRINK,FU
     NNY
```

2.14 Correct Responses

The number of correct responses is tallied using the C variable. After each correct response a one is added to C. After all the words have been presented, the screen is cleared, and the number of problems and the number correct are printed (lines 370, 380 and 390).



```
10  REM  WORD FLASH
20 N = 5
40  FOR J = 1 TO N
50  READ S$(J)
60  NEXT J
70  HOME
90  INPUT "WHAT IS YOUR NAME? ";N
     $
100  FOR K = 1 TO N
110  HOME
140 R =  INT ( RND (1) * N + 1)
150  IF T(R) = 1 THEN 140
160 T(R) = 1
170  VTAB 10
180  HTAB 15
190  PRINT S$(R)
200  FOR D = 1 TO 750
210  NEXT D
220  HOME
230  VTAB 10
240  HTAB 15
250  INPUT A$
260  VTAB 12
270  HTAB 15
280  IF A$ = S$(R) THEN 320
290  PRINT S$(R)
300  GOTO 340
320  PRINT "CORRECT, "N$"!"
330 C = C + 1
340  FOR D = 1 TO 1500
350  NEXT D
360  NEXT K
370  HOME
380  PRINT "NUMBER OF PROBLEMS =
     "N
390  PRINT "NUMBER CORRECT = "C
400  END
410  DATA  ASK,BIG,CLEAN,DRINK,FU
     NNY
```

2.15 More Screen Position

Every item that is printed on the screen should be given careful consideration. Even a very small change might be beneficial. As an example, the CORRECT prompt is printed directly below the word entered. The following change clearly differentiates the feedback following a correct response. Be sure to change line 280 so that control is sent to line 310 after a correct response.

```
10  REM  WORD FLASH
20 N = 5
40  FOR J = 1 TO N
50  READ S$(J)
60  NEXT J
70  HOME
90  INPUT "WHAT IS YOUR NAME? ";N
    $
100  FOR K = 1 TO N
110  HOME
140 R =  INT ( RND (1) * N + 1)
150  IF T(R) = 1 THEN 140
160 T(R) = 1
170  VTAB 10
180  HTAB 15
190  PRINT S$(R)
200  FOR D = 1 TO 750
210  NEXT D
220  HOME
230  VTAB 10
240  HTAB 15
250  INPUT A$
260  VTAB 12
270  HTAB 15
280  IF A$ = S$(R) THEN 310
290  PRINT S$(R)
300  GOTO 340
310  HTAB 12
320  PRINT "CORRECT, "N$"!"
330 C = C + 1
340  FOR D = 1 TO 1500
350  NEXT D
360  NEXT K
370  HOME
380  PRINT "NUMBER OF PROBLEMS =
     "N
390  PRINT "NUMBER CORRECT = "C
400  END
410  DATA  ASK,BIG,CLEAN,DRINK,FU
     NNY
```

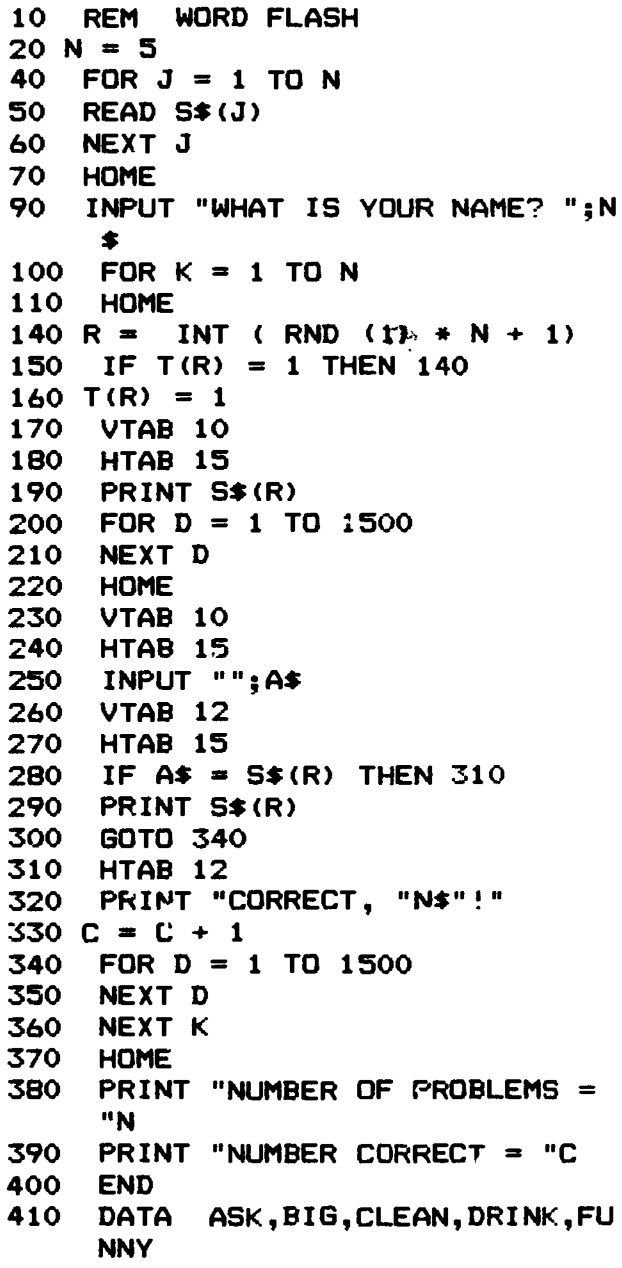
2.16 The INPUT Prompt

The ? prompt generated by the INPUT statement has been suppressed by using a pair of quotation marks, followed by a semicolon.

```
10  REM  WORD FLASH
20 N = 5
40  FOR J = 1 TO N
50  READ S$(J)
60  NEXT J
70  HOME
90  INPUT "WHAT IS YOUR NAME? ";N
     $
100  FOR K = 1 TO N
110  HOME
140 R =  INT ( RND (1) * N + 1)
150  IF T(R) = 1 THEN 140
160 T(R) = 1
170  VTAB 10
180  HTAB 15
190  PRINT S$(R)
200  FOR D = 1 TO 750
210  NEXT D
220  HOME
230  VTAB 10
240  HTAB 15
250  INPUT "";A$
260  VTAB 12
270  HTAB 15
280  IF A$ = S$(R) THEN 310
290  PRINT S$(R)
300  GOTO 340
310  HTAB 12
320  PRINT "CORRECT, "N$"!"
330 C = C + 1
340  FOR D = 1 TO 1500
350  NEXT D
360  NEXT K
370  HOME
380  PRINT "NUMBER OF PROBLEMS =
     "N
390  PRINT "NUMBER CORRECT = "C
400  END
410  DATA  ASK,BIG,CLEAN,DRINK,FU
     NNY
```

2.17 Slow Word Flash Setting

The delay loop in lines 200 and 210 determines the length of time each word appears on the screen. The constant 750 in line 200 causes a delay of approximately one second, and a constant of 1500 in line 200 causes a delay of approximately two seconds.



```
10  REM  WORD FLASH
20 N = 5
40  FOR J = 1 TO N
50  READ S$(J)
60  NEXT J
70  HOME
90  INPUT "WHAT IS YOUR NAME? ";N
     $
100  FOR K = 1 TO N
110  HOME
140 R =  INT ( RND (1) * N + 1)
150  IF T(R) = 1 THEN 140
160 T(R) = 1
170  VTAB 10
180  HTAB 15
190  PRINT S$(R)
200  FOR D = 1 TO 1500
210  NEXT D
220  HOME
230  VTAB 10
240  HTAB 15
250  INPUT "";A$
260  VTAB 12
270  HTAB 15
280  IF A$ = S$(R) THEN 310
290  PRINT S$(R)
300  GOTO 340
310  HTAB 12
320  PRINT "CORRECT, "N$"!"
330 C = C + 1
340  FOR D = 1 TO 1500
350  NEXT D
360  NEXT K
370  HOME
380  PRINT "NUMBER OF PROBLEMS =
     "N
390  PRINT "NUMBER CORRECT = "C
400  END
410  DATA  ASK,BIG,CLEAN,DRINK,FU
     NNY
```

2.18 Fast Word Flash Setting

Setting the constant in line 200 to 250 results in a word presentation on the screen of approximately .33 seconds. In most instances, there is usually no need to present words faster than .2 seconds.

```
10  REM  WORD FLASH
20 N = 5
40  FOR J = 1 TO N
50  READ S$(J)
60  NEXT J
70  HOME
90  INPUT "WHAT IS YOUR NAME? ";N
    $
100  FOR K = 1 TO N
110  HOME
140 R =  INT ( RND (1) * N + 1)
150  IF T(R) = 1 THEN 140
160 T(R) = 1
170  VTAB 10
180  HTAB 15
190  PRINT S$(R)
200  FOR D = 1 TO 250
210  NEXT D
220  HOME
230  VTAB 10
240  HTAB 15
250  INPUT "";A$
260  VTAB 12
270  HTAB 15
280  IF A$ = S$(R) THEN 310
290  PRINT S$(R)
300  GOTO 340
310  HTAB 12
320  PRINT "CORRECT, "N$"!"
330 C = C + 1
340  FOR D = 1 TO 1500
350  NEXT D
360  NEXT K
370  HOME
380  PRINT "NUMBER OF PROBLEMS =
     "N
390  PRINT "NUMBER CORRECT = "C
400  END
410  DATA  ASK,BIG,CLEAN,DRINK,FU
     NNY
```

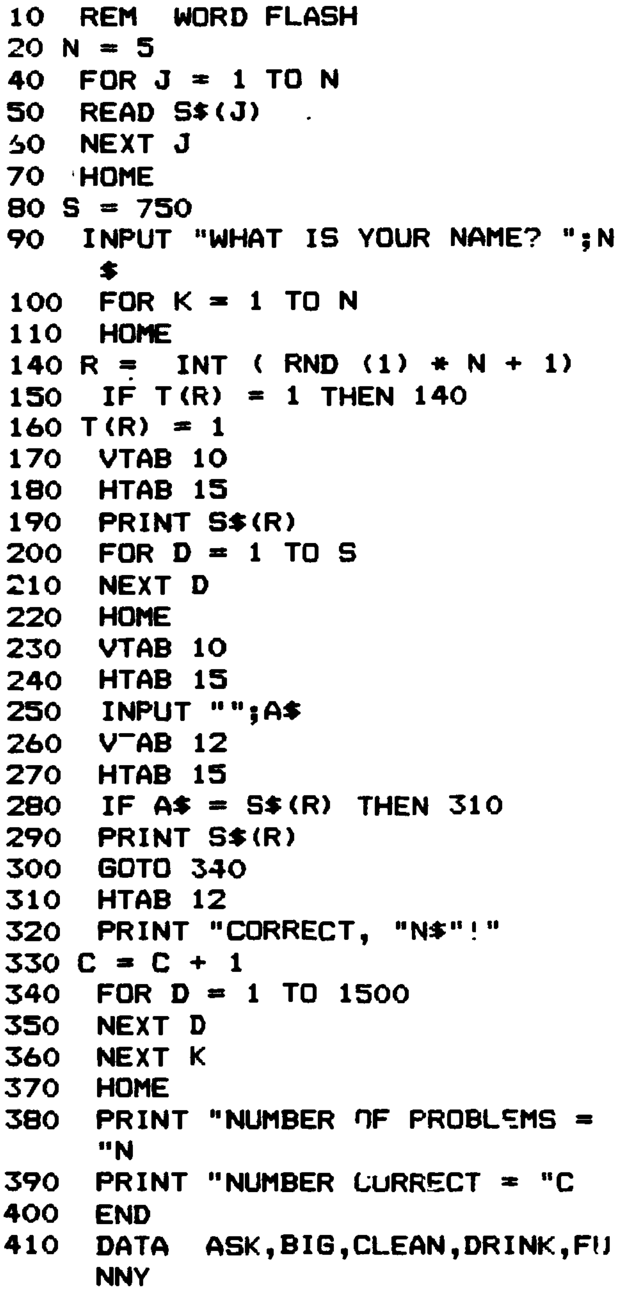
2.19 INPUT Speed Setting

Instead of specifying the speed in line 200, the variable S can be set using an INPUT statement prior to running the program. In addition to the added INPUT statement in line 80, the variable S must also be added to the loop beginning in line 200.

```
10  REM  WORD FLASH
20 N = 5
40  FOR J = 1 TO N
50  READ S$(J)
60  NEXT J
70  HOME
80  INPUT "ENTER SPEED: ";S
90  INPUT "WHAT IS YOUR NAME? ";N
     $
100  FOR K = 1 TO N
110  HOME
140 R =  INT ( RND (1) * N + 1)
150  IF T(R) = 1 THEN 140
160 T(R) = 1
170  VTAB 10
180  HTAB 15
190  PRINT S$(R)
200  FOR D = 1 TO S
210  NEXT D
220  HOME
230  VTAB 10
240  HTAB 15
250  INPUT "";A$
260  VTAB 12
270  HTAB 15
280  IF A$ = S$(R) THEN 310
290  PRINT S$(R)
300  GOTO 340
310  HTAB 12
320  PRINT "CORRECT, "N$"!"
330 C = C + 1
340  FOR D = 1 TO 1500
350  NEXT D
360  NEXT K
370  HOME
380  PRINT "NUMBER OF PROBLEMS =
     "N
390  PRINT "NUMBER CORRECT = "C
400  END
410  DATA  ASK,BIG,CLEAN,DRINK,FU
     NNY
```

2.20 Variable Speed Setting

The speed can also be set by assigning a value to S in line 80.



```
10  REM  WORD FLASH
20 N = 5
40  FOR J = 1 TO N
50  READ S$(J)
60  NEXT J
70  HOME
80 S = 750
90  INPUT "WHAT IS YOUR NAME? ";N
    $
100  FOR K = 1 TO N
110  HOME
140 R =  INT ( RND (1) * N + 1)
150  IF T(R) = 1 THEN 140
160 T(R) = 1
170  VTAB 10
180  HTAB 15
190  PRINT S$(R)
200  FOR D = 1 TO S
210  NEXT D
220  HOME
230  VTAB 10
240  HTAB 15
250  INPUT "";A$
260  V¯AB 12
270  HTAB 15
280  IF A$ = S$(R) THEN 310
290  PRINT S$(R)
300  GOTO 340
310  HTAB 12
320  PRINT "CORRECT, "N$"!"
330 C = C + 1
340  FOR D = 1 TO 1500
350  NEXT D
360  NEXT K
370  HOME
380  PRINT "NUMBER OF PROBLEMS =
     "N
390  PRINT "NUMBER CORRECT = "C
400  END
410  DATA  ASK,BIG,CLEAN,DRINK,FU
     NNY
```

2.21 Adding Words

More words can be included in each run by changing the value of N in line 20, and then adding a corresponding number of words. If N is 10, then there should be at least 10 words in DATA statements. There can be more words than N, but fewer words will result in an error.

```
10  REM  WORD FLASH
20 N = 10
40  FOR J = 1 TO N
50  READ S$(J)
60  NEXT J
70  HOME
80 S = 750
90  INPUT "WHAT IS YOUR NAME? ";N
    $
100  FOR K = 1 TO N
110  HOME
140 R =  INT ( RND (1) * N + 1)
150  IF T(R) = 1 THEN 140
160 T(R) = 1
170  VTAB 10
180  HTAB 15
190  PRINT S$(R)
200  FOR D = 1 TO S
210  NEXT D
220  HOME
230  VTAB 10
240  HTAB 15
250  INPUT "";A$
260  VTAB 12
270  HTAB 15
280  IF A$ = S$(R) THEN 310
290  PRINT S$(R)
300  GOTO 340
310  HTAB 12
320  PRINT "CORRECT, "N$"!"
330 C = C + 1
340  FOR D = 1 TO 1500
350  NEXT D
360  NEXT K
370  HOME
380  PRINT "NUMBER OF PROBLEMS =
     "N
390  PRINT "NUMBER CORRECT = "C
400  END
410  DATA  ASK,BIG,CLEAN,DRINK,FU
     NNY
420  DATA  GOOD-BYE,HELP,LITTLE,M
     AKE,NO
```

2.22 The DIM Statement

If more than 10 words are used, space must be reserved in the Apple's memory for the arrays used (line 30).

```
10  REM  WORD FLASH
20 N = 15
30  DIM S$(N),T(N)
40  FOR J = 1 TO N
50  READ S$(J)
60  NEXT J
70  HOME
80 S = 750
90  INPUT "WHAT IS YOUR NAME? ";N
     $
100  FOR K = 1 TO N
110  HOME
140 R =  INT ( RND (1) * N + 1)
150  IF T(R) = 1 THEN 140
160 T(R) = 1
170  VTAB 10
180  HTAB 15
190  PRINT S$(R)
200  FOR D = 1 TO S
210  NEXT D
220  HOME
230  VTAB 10
240  HTAB 15
250  INPUT "";A$
260  VTAB 12
270  HTAB 15
280  IF A$ = S$(R) THEN 310
290  PRINT S$(R)
300  GOTO 340
310  HTAB 12
320  PRINT "CORRECT, "N$"!"
330 C = C + 1
340  FOR D = 1 TO 1500
350  NEXT D
360  NEXT K
370  HOME
380  PRINT "NUMBER OF PROBLEMS =
     "N
390  PRINT "NUMBER CORRECT = "C
400  END
410  DATA  ASK,BIG,CLEAN,DRINK,FU
     NNY
420  DATA  GOOD-BYE,HELP,LITTLE,M
     AKE,NO
430  DATA  NOW,PLEASE,TODAY,TRY,Y
     ES
```

2.23 A Final Delay

The delay loop in lines 120 and 130 results in a brief pause (with a clear screen) between problems.

```
10  REM  WORD FLASH
20 N = 15
30  DIM S$(N),T(N)
40  FOR J = 1 TO N
50  READ S$(J)
60  NEXT J
70  HOME
80 S = 750
90  INPUT "WHAT IS YOUR NAME? ";N
    $
100  FOR K = 1 TO N
110  HOME
120  FOR D = 1 TO 750
130  NEXT D
140 R =  INT ( RND (1) * N + 1)
150  IF T(R) = 1 THEN 140
160 T(R) = 1
170  VTAB 10
180  HTAB 15
190  PRINT S$(R)
200  FOR D = 1 TO S
210  NEXT D
220  HOME
230  VTAB 10
240  HTAB 15
250  INPUT "";A$
260  VTAB 12
270  HTAB 15
280  IF A$ = S$(R) THEN 310
290  PRINT S$(R)
300  GOTO 340
310  HTAB 12
320  PRINT "CORRECT, "N$"!"
330 C = C + 1
340  FOR D = 1 TO 1500
350  NEXT D
360  NEXT K
370  HOME
380  PRINT "NUMBER OF PROBLEMS =
     "N
390  PRINT "NUMBER CORRECT = "C
400  END
410  DATA  ASK,BIG,CLEAN,DRINK,FU
     NNY
420  DATA  GOOD-BYE,HELP,LITTLE,M
     AKE,NO
430  DATA  NOW,PLEASE,TODAY,TRY,Y
     ES
```

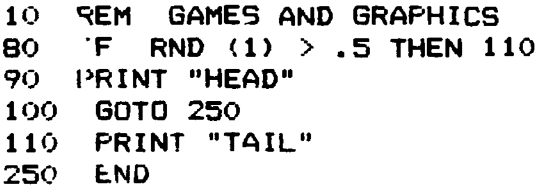
3.0 Games and Graphics

Game formats and graphic techniques can be used to enhance learning and as motivational devices. The Games and Graphics program in this section is designed to illustrate two fundamental concepts in the design of games and graphic presentation:

RANDOM SIMULATION

GRAPHIC MOVEMENT

The initial concept presented below demonstrates a simple "flip" of the coin. A random number is generated in line 80. If the number is greater than .5, then control is transferred to line 110 and TAIL is printed; if the number generated is equal to or less than .5, then line 90 is processed and HEAD is printed.



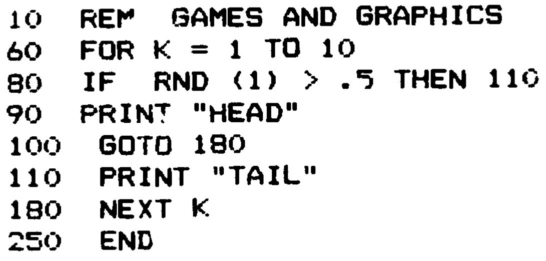
```
10  REM  GAMES AND GRAPHICS
80  IF  RND (1) > .5 THEN 110
90  PRINT "HEAD"
100  GOTO 250
110  PRINT "TAIL"
250  END
```

3.1 Loops

The kernel program is first modified to "flip" the coin ten times by adding the loop in lines 60 and 180. In this loop, lines 80 through 110 are processed (or executed) a total of ten times.

```
60 FOR K = 1 TO 10
180 NEXT K
```

Notice also that line 100 is changed so that control is sent to line 180 rather than line 250. What would happen is line 100 is not changed?

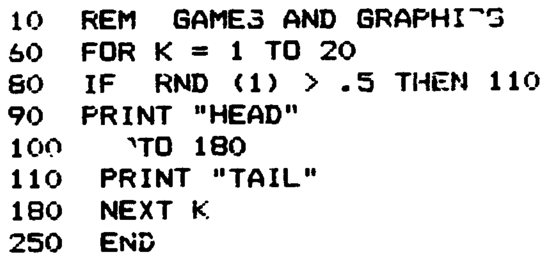


```
10  REM  GAMES AND GRAPHICS
60  FOR K = 1 TO 10
80  IF  RND (1) > .5 THEN 110
90  PRINT "HEAD"
100  GOTO 180
110  PRINT "TAIL"
180  NEXT K
250  END
```

3.2 Loop Size

The size of a loop indicates the number of iterations or the number of times a loop is processed. The last value in a loop signifies the maximum number of times a loop is processed.

Instead of flipping our Apple coin ten times, we can easily have the ever-obedient Apple flip the coin 100 times (or 1000 for that matter). The listing below shows how the coin is flipped a mere 20 times.



```
10  REM  GAMES AND GRAPHICS
60  FOR K = 1 TO 20
80  IF  RND (1) > .5 THEN 110
90  PRINT "HEAD"
100   GOTO 180
110  PRINT "TAIL"
180  NEXT K
250  END
```

What change is necessary to flip the coin 100 times?

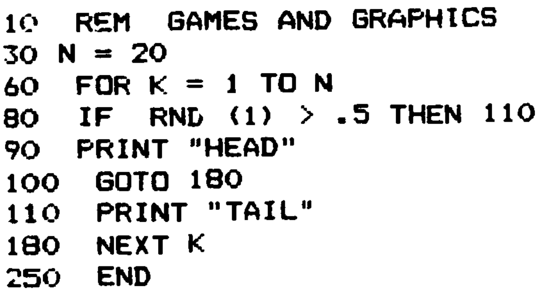
3.3 N for Iterations

Rather than changing the maximum number of loop iterations in line 60 each time we want to flip a different number of coins, we can first specify the number of iterations in variable N and then using this variable in the beginning of the loop.

```
30 N = 20

60 FOR K = 1 TO N
```

Before going on to the next modification, run the program using N sizes of 5 and 100. Were you able to keep count of the heads and tails when N was set to 100? Do you have any ideas for keeping track of the numbers of heads and tails?



```
10  REM  GAMES AND GRAPHICS
30 N = 20
60  FOR K = 1 TO N
80  IF  RND (1) > .5 THEN 110
90  PRINT "HEAD"
100  GOTO 180
110  PRINT "TAIL"
180  NEXT K
250  END
```

3.4 HOME

Before we go on to the problem of keeping track of the number of heads and tails, it's a good idea to insert a HOME statement to clear the screen when we first run the program. Now, each time the program is run, the screen is cleared and the cursor sent to the upper left-hand corner of the screen.

```
10  REM  GAMES AND GRAPHICS
20  HOME
30 N = 20
60  FOR K = 1 TO N
80  IF  RND (1) > .5 THEN 110
90  PRINT "HEAD"
100  GOTO 180
110  PRINT "TAIL"
180  NEXT K
250  END
```

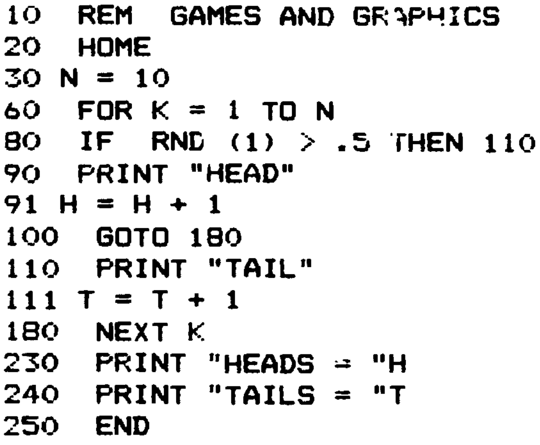
3.5 The Number of Heads and Tails

There are several methods for keeping track of the number of heads and tails. One technique is to use variables to count the number of heads and tails after each toss. This is accomplished by using the variale H to track the number of heads and T to track the number of tails.

After each toss, a 1 is added to either H or T. Remember that when you first run the program, all variables (including H and T) are set to 0.

```
91 H = H + 1
111 T = T + 1
```

After N tosses, lines 181 amd 182 print the total number of heads and tails for a given run.



```
10  REM  GAMES AND GRAPHICS
20  HOME
30 N = 10
60  FOR K = 1 TO N
80  IF  RND (1) > .5 THEN 110
90  PRINT "HEAD"
91 H = H + 1
100  GOTO 180
110  PRINT "TAIL"
111 T = T + 1
180  NEXT K
230  PRINT "HEADS = "H
240  PRINT "TAILS = "T
250  END
```

3.6 Random Integers

A second technique that can be used to determine whether a coin flip is a heads or tails entails generating a random integer. This modification becomes extremely important as we make continued game and graphics modifications.

We first delete line 80 and then add lines 70 and 85.

```
80

70 R = INT ( RND(1) * 2)

85 IF R = 1 THEN 110
```

Line 70 generates a random number of either 0 or 1. If the number generated and stored in R is 1, then control is sent to line 110 (tails); if not, line 90 is processed (heads).

```
10  REM  GAMES AND GRAPHICS
20  HOME
30 N = 10
60  FOR K = 1 TO N
70 R =  INT ( RND (1) * 2)
85  IF R = 1 THEN 110
90  PRINT "HEAD"
91 H = H + 1
100  GOTO 180
110  PRINT "TAIL"
111 T = T + 1
180  NEXT K
230  PRINT "HEADS = "H
240  PRINT "TAILS = "T
250  END
```

3.7 Arrays

With the addition of the random integer in variable R, it is an easy matter to use an array to keep track of the number of heads and tails. The heads will be tallied by using P(0) and the tails by using P(1). Thus, each time a head is flipped, a 1 is added to P(0); and each time a tail is flipped, a 1 is added to P(1).

Remember that when the program is first run, all variables (and array elements) are set to 0. Also, be sure to change lines 230 and 240 so that the contents of P(0) and P(1) are printed at the end of the run.

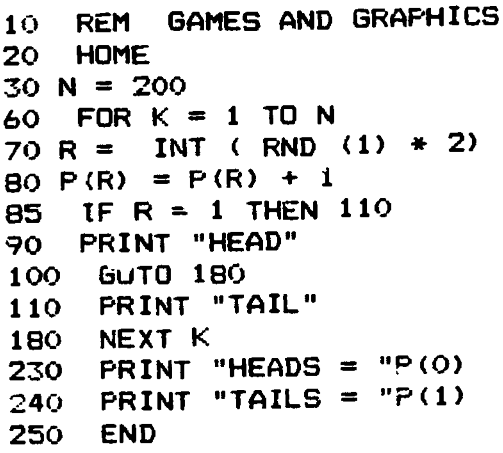
```
10  REM  PROBABILITY GAMES
20  HOME
30 N = 20
60  FOR K = 1 TO N
70 R =  INT ( RND (1) * 2)
85  IF R = 1 THEN 110
90  PRINT "HEAD"
91 P(R) = P(R) + 1
100  GOTO 180
110  PRINT "TAIL"
111 P(R) = P(R) + 1
180  NEXT K
230  PRINT "HEADS = "P(0)
240  PRINT "TAILS = "P(')
250  END
```

3.8 Shortcut Count

The advantage of arrays is that they frequently provide a quicker means to achieve a desired programming result. As an example, instead of using two different lines to tally the number of heads and tails (lines (91 and 111), these lines could be deleted and just one line used to tack heads and tails.

```
80 P(R) = P(R) + 1
91
111
```

In line 80, if R is 0, then 1 is added to P(0); and if R is 1, then 1 is added to P(1).



```
10  REM  GAMES AND GRAPHICS
20  HOME
30 N = 200
60  FOR K = 1 TO N
70 R =  INT ( RND (1) * 2)
80 P(R) = P(R) + 1
85  IF R = 1 THEN 110
90  PRINT "HEAD"
100  GOTO 180
110  PRINT "TAIL"
180  NEXT K
230  PRINT "HEADS = "P(0)
240  PRINT "TAILS = "P(1)
250  END
```

3.9 More about Loops

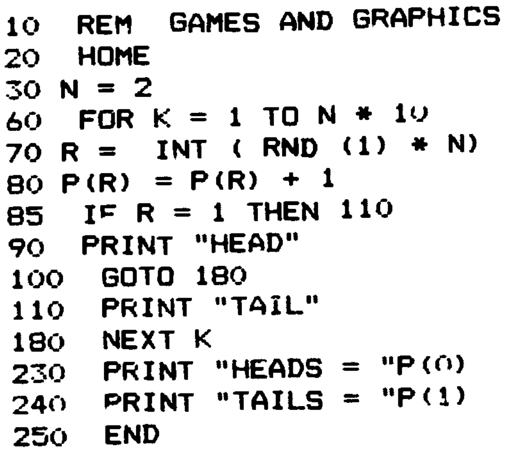
Thus far the variable N has been used to indicate the number of iterations for a loop. The number of iterations for a loop can also be adjusted by modifying the FOR statement in line 60. As shown below, N in line 60 is multiplied by 10. The asterisk (*) is used to signify multiplication in BASIC

```
60 FOR K = 1 TO N * 10
```

If N is set to 2 in line 30, then line 60 begins a loop that has 20 iterations.

```
30 N = 2
```

Because N is now 2, we can substitute N for 2 in line 70 to generate a random number of 0 or 1.



```
10  REM  GAMES AND GRAPHICS
20  HOME
30 N = 2
60  FOR K = 1 TO N * 10
70 R =  INT ( RND (1) * N)
80 P(R) = P(R) + 1
85  IF R = 1 THEN 110
90  PRINT "HEAD"
100  GOTO 180
110  PRINT "TAIL"
180  NEXT K
230  PRINT "HEADS = "P(0)
240  PRINT "TAILS = "P(1)
250  END
```

3.10 Multiple Outcomes

We need not restrict our simulation to heads or tails. By changing line N we can specify whatever number of outcomes desired. If we wanted to simulate the toss of a die, N would be set to 6 which indicates that the random number generated in line 70 and stored in R will be a 0, 1, 2, 3, 4, or 5.

Because we are now specifying six possible outcomes, lines 85, 90, 100, and 110 are deleted. The loop in lines 200, 210 and 220 has beed added to print the results.

```
10  REM  GAMES AND GRAPHICS
20  HOME
30 N = 6
60  FOR K = 1 TO N * 10
70 R =  INT ( RND (1) * N)
80 P(R) = P(R) + 1
180  NEXT K
200  FOR K = 1 TO N
210  PRINT K" "P(K)
220  NEXT K
250  END
```

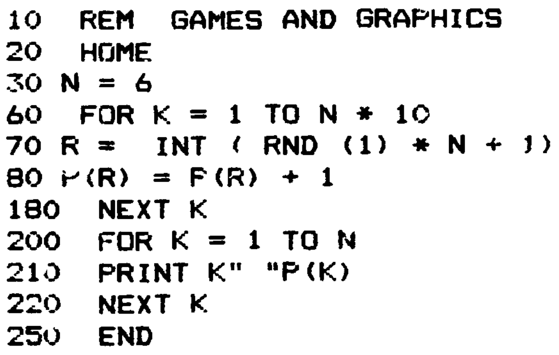
Run this program five times. What is wrong with the results? Why is the number 6 always 0?

3.11 Random Numbers from 1 to 6

The problem with the last modification resulted from the fact that random numbers from 0 to 5 were generated. This, of course, is indeed six numbers, but they are six numbers stored in array locations from P(0) to P(5). The loop in line 200 prints array locations from P(1) to P(6). This is so because K in line 200 varies from 1 to N or six

In short, array location P(0) was not printed, and memory location P(6), which was empty, was printed.

What we need to get everything back in sync is a random number generated in line 70 that can be a 1, 2, 3, 4, 5 or 6. Because the random number generated now varies between 0 and 5, a number between 1 and 6 is produced by adding 1.



```
10  REM  GAMES AND GRAPHICS
20  HOME
30 N = 6
60  FOR K = 1  TO N * 10
70 R =  INT ( RND (1) * N + 1)
80 P(R) = P(R) + 1
180  NEXT K
200  FOR K = 1 TO N
210  PRINT K" "P(K)
220  NEXT K
250  END
```

3.12 Loop Size

In the last two modifications, the number of random numbers generated was N * 10 or 60 because N was equal to 6.

The number of random numbers generated is easily increased to N * 25 or 150 by changing the constant in line 60.

```
10  REM  GAMES AND GRAPHICS
20  HOME
30 N = 6
60  FOR K = 1 TO N * 25
70 R =  INT ( RND (1) * N + 1)
80 P(R) = P(R) + 1
100  NEXT K
200  FOR K = 1 TO N
210  PRINT P(K)
220  NEXT K
250  END
```

3.13 The INPUT Statement

An INPUT statement can be used in line 30 to avoid changing the BASIC program each time N is varied. Each time the program is run with this modification, the prompt ENTER NUMBER is printed.

```
30 INPUT "ENTER NUMBER: ";N
```

Be sure to include the semicolon after the last parenthesis in line 30 or a syntax error will result.

```
10  REM  GAMES AND GRAPHICS
20  HOME
30  INPUT "ENTER NUMBER: ";N
60  FOR K = 1 TO N * 25
70 R =  INT ( RND (1) * N + 1)
80 P(R) = P(R) + 1
180  NEXT K
200  FOR K = 1 TO N
210  PRINT P(K)
220  NEXT K
250  END
```

3.14 Checking Input

What would happen if a student entered -4 or 5325 after the ENTER NUMBER prompt? As the program is now written, an error would result.

An IF statement can be added to the program to insure that a number within a specified range is entered. As shown below, if the number entered and stored in N is less than 2 or greater than 9, then control is sent back to line 20, the screen is cleared, and a new number is entered.

```
10  REM  GAMES AND GRAPHICS
20  HOME
30  INPUT "ENTER NUMBER: ';N
40  IF N < 2 OR N > 9 THEN 20
60  FOR K = 1 TO N * 25
70 R =  INT ( RND (1) * N + 1)
80 P(R) = P(R) + 1
180  NEXT K
200  FOR K = 1 TO N
210  PRINT P(K)
220  NEXT K
250  END
```

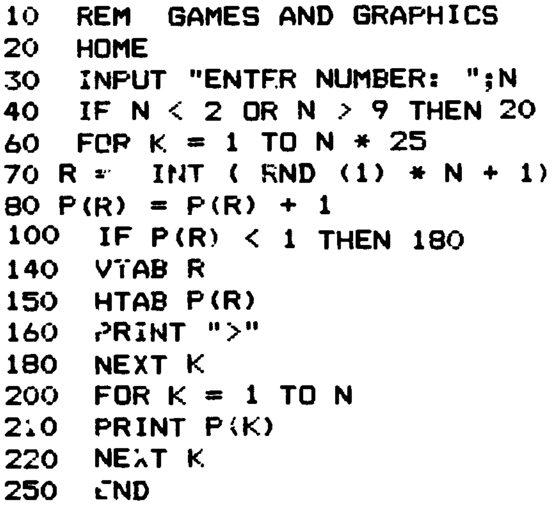
3.15 Movement

Most arcade games have some type of graphic movement. The addition of lines 90 through 170 demonstrate how a screen character can be moved across the screen...sort of.

The key to this modification is lines 140, 150 and 160:

```
140 VTAB R
150 HTAB P(R)
160 PRINT " "
```

The VTAB gives each of the possible random numbers specified a row position, and the values in P(R) are used to specify a column position. As an example, each time a 1 is added to P(4), a character is printed at the screen location designated by row position VTAB 4 and column position P(4). If the value in P(4) is 12, then the column position is HTAB 12.



```
10  REM  GAMES AND GRAPHICS
20  HOME
30  INPUT "ENTER NUMBER: ";N
40  IF N < 2 OR N > 9 THEN 20
60  FOR K = 1 TO N * 25
70 R =  INT ( RND (1) * N + 1)
80 P(R) = P(R) + 1
100  IF P(R) < 1 THEN 180
140  VTAB R
150  HTAB P(R)
160  PRINT ">"
180  NEXT K
200  FOR K = 1 TO N
210  PRINT P(K)
220  NEXT K
250  END
```

3.16 Screen Cleaning

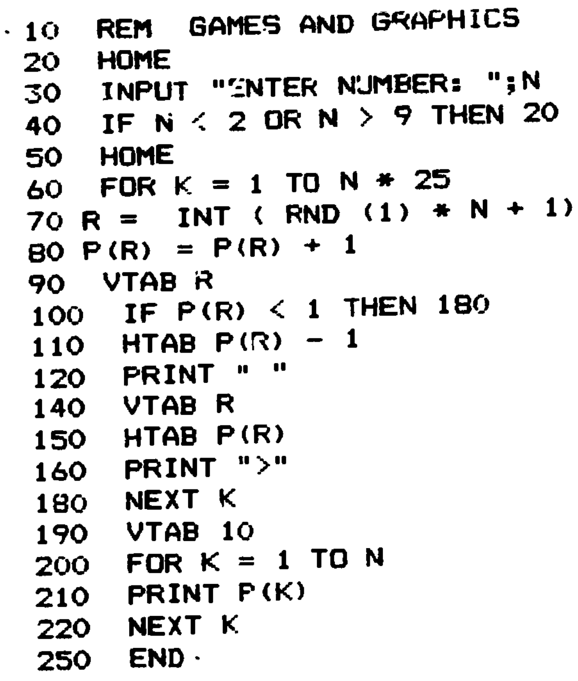
When the program is run, the character moves across the screen prompt ENTER NUMBER, and then the numeric results are printed over the screen characters. This rather messy situation is easily rectified by adding lines 50 and 190.

```
10  REM  GAMES AND GRAPHICS
20  HOME
30  INPUT "ENTER NUMBER: ";N
40  IF N < 2 OR N > 9 THEN 20
50  HOME
60  FOR K = 1 TO N * 25
70 R =  INT ( RND (1) * N + 1)
80 P(R) = P(R) + 1
100  IF P(R) < 1 THEN 180
140  VTAB R
150  HTAB P(R)
160  PRINT ">"
180  NEXT K
190  VTAB 10
200  FOR  K = 1 TO N
210  PRINT P(K)
220  NEXT K
250  END
```

3.17 The Illusion of Movement

Screen movement is a video illustion. A shape or character does not actually "move" across the screen. What happens is that the same shape or character is printed at different screen locations. The illusion of movement is created by flashing the identical shape or character at consecutive screen locations.

The additions in lines 90, 110 and 120 erases the last character printed before the same character is printed at the next consecutive screen location.



```
10  REM  GAMES AND GRAPHICS
20  HOME
30  INPUT "ENTER NUMBER: ";N
40  IF N < 2 OR N > 9 THEN 20
50  HOME
60  FOR K = 1 TO N * 25
70 R =  INT ( RND (1) * N + 1)
80 P(R) = P(R) + 1
90  VTAB R
100  IF P(R) < 1 THEN 180
110  HTAB P(R) - 1
120  PRINT " "
140  VTAB R
150  HTAB P(R)
160  PRINT ">"
180  NEXT K
190  VTAB 10
200  FOR K = 1 TO N
210  PRINT P(K)
220  NEXT K
250  END
```

3.18 A Moving Star

Our rather mundane arrow is easily replaced by an asterisk that, with a little imagination, can be viewed as star. If something other than a star is desired, line 160 can be changed to print (and move) any variety of shapes, characters or combination of characters.

```
10  REM  GAMES AND GRAPHICS
20  HOME
30   INPUT "ENTER NUMBER: ";N
40   IF N < 2 OR N > 9 THEN 20
50  HOME
60  FOR K = 1 TO N * 25
70 R =  INT ( RND (1) * N + 1)
80 P(R) = P(R) + 1
90  VTAB R
100  IF P(R) < 1 THEN 180
110  HTAB P(R) - 1
120  PRINT " "
140  VTAB R
150  HTAB P(R)
160  PRINT "*"
180  NEXT K
190  VTAB 10
200  FOR K = 1 TO N
210  PRINT P(K)
220  NEXT K
250  END
```

3.19 Flashing Stars

The FLASH mode is activated just prior to printing by adding line 130. Immediately after the asterisk is printed and flashed, the video mode is set to normal. What happens if the video mode is not set to normal in line 170?

```
10  REM  GAMES AND GRAPHICS
20  HOME
30  INPUT "ENTER NUMBER: ";N
40  IF N < 2 OR N > 9 THEN 20
50  HOME
60  FOR K = 1 TO N * 25
70 R =  INT ( RND (1) * N + 1)
80 P(R) = P(R) + 1
90  VTAB R
100  IF P(R) < 1 THEN 180
110  HTAB P(R) - 1
120  PRINT " "
130  FLASH
140  VTAB R
150  HTAB P(R)
160  PRINT "*"
170  NORMAL
180  NEXT K
190  VTAB 10
200  FOR K = 1 TO N
210  PRINT P(K)
220  NEXT K
250  END
```

3.20 A Winner

A winner in our great star race is determined by restructuring the loop in lines 200, 210, 220 and 230. Note that when the first comparison is made in line 220, W is 0. If the value in P(K) is greater than the value in P(W), which is 0, then K is inserted in W.

Given a tie between 1, 2 and 3, which would be declared the winner as a result of the comparisons made in line 220?

```
10  REM  GAMES AND GRAPHICS
20  HOME
30  INPUT "ENTER NUMBER: ";N
40  IF N < 2 OR N > 9 THEN 20
50  HOME
60  FOR K = 1 TO N * 25
70 R =  INT ( RND (1) * N + 1)
80 P(R) = P(R) + 1
90  VTAB R
100  IF P(R) < 1 THEN 180
110  HTAB P(R) - 1
120  PRINT " "
130  FLASH
140  VTAB R
150  HTAB P(R)
160  PRINT "*"
170  NORMAL
180  NEXT K
190  VTAB 10
200  FOR K = 1 TO N
210  PRINT "STAR SPEED = "P(K)
220  IF P(K) > P(W) THEN W = K
230  NEXT K
240  PRINT "THE WINNER IS #"W
250  END
```

3.21 Game Development

The key to every game is the game scenario. We can give a little life to our racing stars to something a little more unique, like racing camels (why not?).

The new game concept is accomplished by setting N to 3 (or whatever) in line 30, and then adding names to the N$ array. Lines 160 and 210 are also modified to print the names in the N$ array.

```
10  REM  GAMES AND GRAPHICS
20  HOME
30 N = 3
40 N$(1) = "FLIP"
41 N$(2) = "FLAP"
42 N$(3) = "FLOP"
50  HOME
60  FOR K = 1 TO N * 25
70 R =  INT ( RND (1) * N + 1)
80 P(R) = P(R) + 1
90  VTAB R
100  IF P(R) < 1 THEN 180
110  HTAB P(R) - 1
120  PRINT " "
130  FLASH
140  VTAB R
150  HTAB P(R)
160  PRINT N$(R)
170  NORMAL
180  NEXT K
190  VTAB 10
200  FOR K = 1 TO N
210  PRINT "CAMEL AND SPEED: "N$(
     K)" "P(K)
220  IF P(K) > P(W) THEN W = K
230  NEXT K
240  PRINT "THE WINNER IS #"W
250  END
```

3.22 A Mini-Arcade Game

Several additional modifications can be made to produce a shooting target game. The object is to press the RETURN key just as the asterisk is over the ^ character. A point is rewarded for every hit.

The PEEK function in line 180 determines when a key is pressed. In line 220, if you pressed a key and the asterisk was in the 20th column position (which is over the ^ character), a 1 is added to H which tallies the number of hits.

```
10  REM  GAMES AND GRAPHICS
20  HOME
30 N = 1
40  FOR L = 1 TO 10
50  HOME
51  VTAB 3
52  HTAB 20
53  PRINT "^"
54  VTAB 5
55  HTAB 16
56  PRINT "HITS = "H
60  FOR K = 1 TO N * 25
70 R = 1
80 P(R) = P(R) + 1
90  VTAB R
100  IF P(R) < 1 THEN 180
110  HTAB P(R) - 1
120  PRINT " "
130  FLASH
140  VTAB R
150  HTAB P(R)
160  PRINT "*"
170  NORMAL
180 X =  PEEK ( - 16384)
190  IF X > 127 THEN 220
200  NEXT K
210  GOTO 240
220  IF K = 20 THEN H = H + 1
221  POKE  - 16368,0
230 P(R) = 0
240  NEXT L
250  END
```

3.23 Falling from the Sky

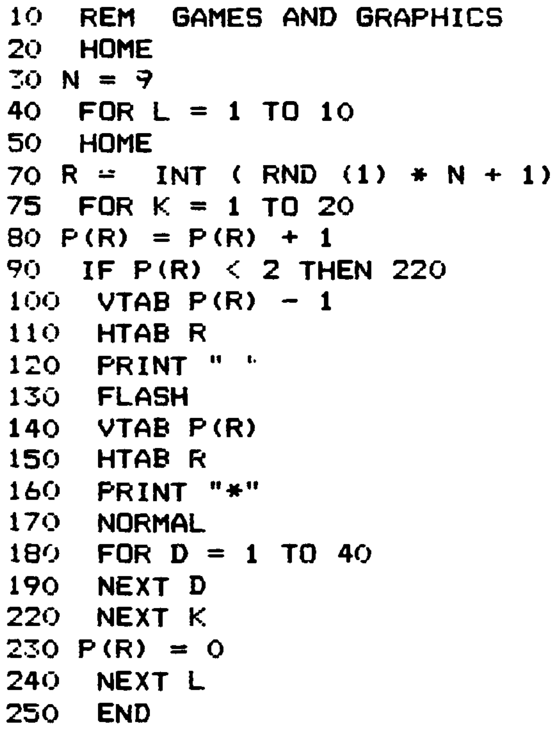
Everyone knows that sh · and characters falling from the sky is an integral part of many arcade games. Instead of stars moving horizontally, we can redirect the movement to a vertical format by reversing the variables used to indicate row and column position in the VTAB abd HTAB statements.

As an example, lines 140 and 150 are changed from

```
140 VTAB R
150 HTAB P(R)
```

to

```
140 VTAB P(R)
150 HTAB R
```



```
10  REM  GAMES AND GRAPHICS
20  HOME
30 N = 9
40  FOR L = 1 TO 10
50  HOME
70 R =  INT ( RND (1) * N + 1)
75  FOR K = 1 TO 20
80 P(R) = P(R) + 1
90  IF P(R) < 2 THEN 220
100  VTAB P(R) - 1
110  HTAB R
120  PRINT " "
130  FLASH
140  VTAB P(R)
150  HTAB R
160  PRINT "*"
170  NORMAL
180  FOR D = 1 TO 40
190  NEXT D
220  NEXT K
230 P(R) = 0
240  NEXT L
250  END
```