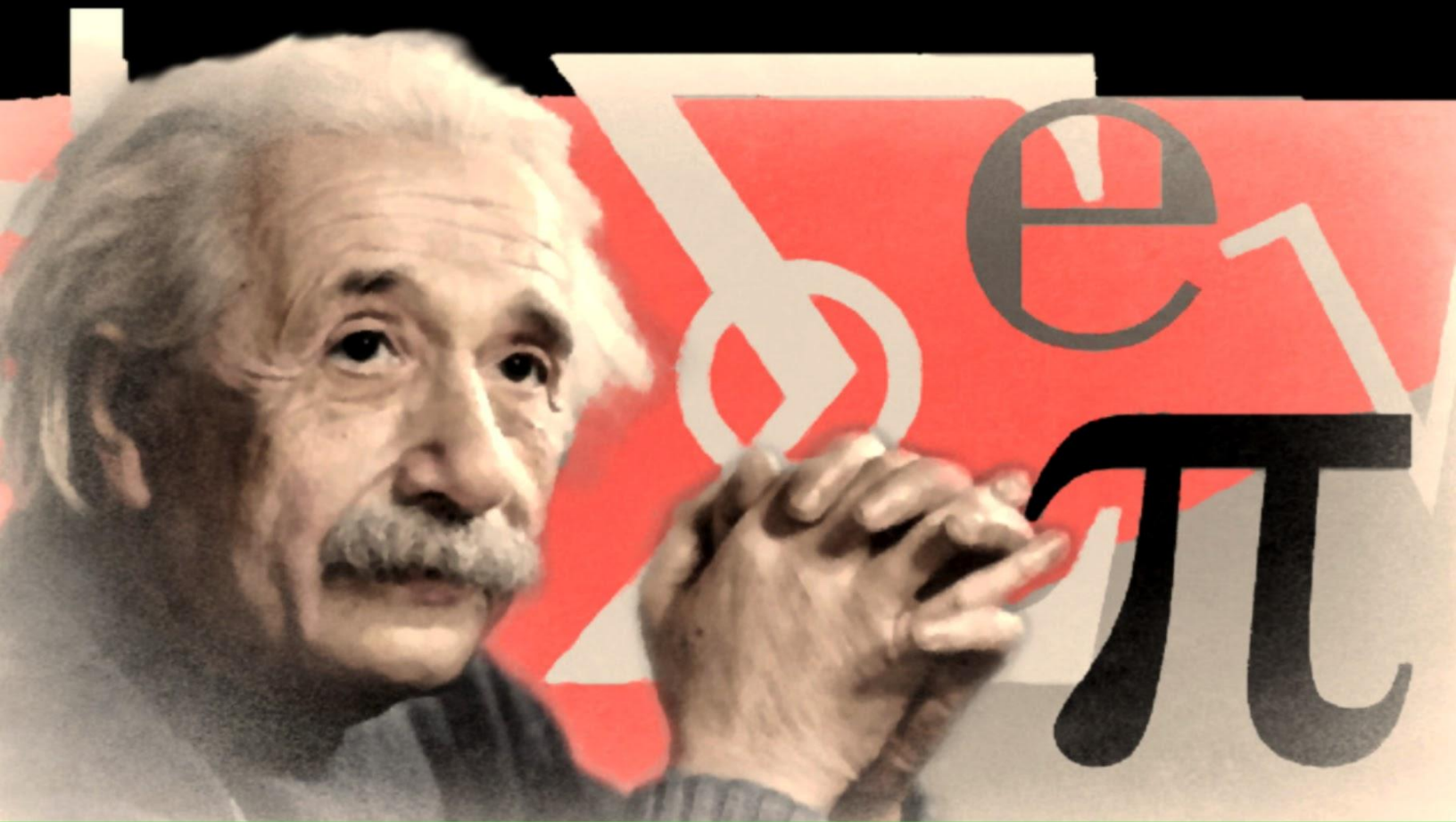


ATARI EDUCATIONAL SYSTEM
BASIC Applications Programs - Mathematics I



BASIC Application Programs

Mathematics I

The programs in this booklet were originally printed in a 1972 publication from Digital Equipment Corporation's Education Department titled **BASIC Application Programs - Mathematics I**.

The programs contained in this series are designed to demonstrate how the computer can be applied in a meaningful way to problems of many disciplines. The problems and the corresponding programs are, for the most part, quite simple and are designed to be "jumping off points" for students from high school level on up.

The programs have been modified from DEC BASIC to the Atari Microsoft BASIC II programming language. As the programs were originally designed for wide, continuous-feed line printers, they have been adjusted to fit the 40 column wide by 24 lines high text windows (GRAPHICS MODE 0) on the Atari 8-bit home computer.

All of the programs should run on any Atari 8-bit home computer or emulator using Atari Microsoft BASIC II programming language.

Contents

Chapter	Title	Page
Chapter One	Exponential Convergence	4
Chapter Two	Calculator	6
Chapter Three	Computer Assisted Instruction	9
Chapter Four	Series Convergence	14
Chapter Five	Extended Calculator	18
Chapter Six	ROOTS	20
Chapter Seven	SETS	23
Chapter Eight	Simultaneous Equations	26
Chapter Nine	Area	29
Chapter Ten	Pythagorean Theorem	32
Chapter Eleven	Guess	35
Chapter Twelve	Gambling Simulation	37
Chapter Thirteen	Patterns	40
Chapter Fourteen	Plotting	43

Chapter One

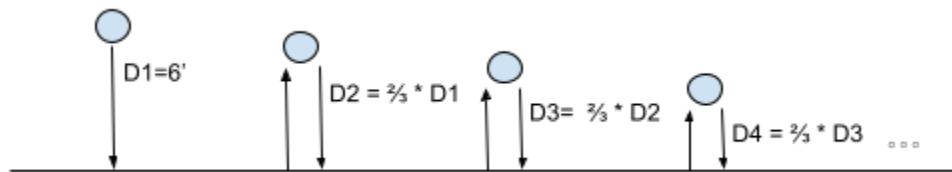
Exponential Convergence

This program illustrates how the computer can be used to solve a problem using exponential convergence.

The problem is:

Holding a ball six feet from the floor and letting it drop, a basketball player noticed that the ball bounced back up only two-thirds of that height (four feet). On the second bounce, the ball rose to only two thirds of the second height (two feet, eight inches). The third bound brought it to two-thirds of the previous height, and so on. What was the total distance the ball traveled--both up and down--before it came to rest?

A diagram can be used to show how far the ball bounced each time:



Notice that after the first bounce, the ball has traveled (at floor level):

$$D1 + 2 * D2 \quad \text{or} \quad D1 + 2 * (\frac{2}{3} * D1)$$

and that after two bounces, the ball has traveled:

$$D1 + 2 * D2 + 2 * D3 = D1 + 2 * (\frac{2}{3} * D1) + 2 * (\frac{2}{3} * \frac{2}{3} * D1)$$

Hence, with each additional bounce, the ball travels an additional distance of:

$$2 * D1 * (\frac{2}{3})^n \quad \text{or} \quad 12 * (\frac{2}{3})^n$$

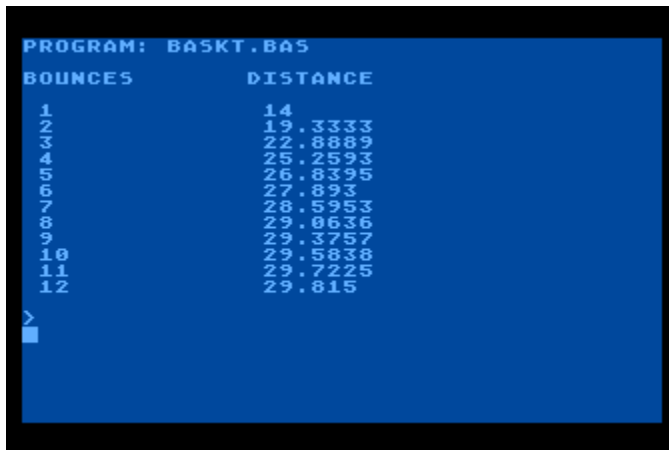
Now it is a simple matter to write a program which increments the total distance traveled and print out the results after each bounce.

```
10 ' SAVE "D1:BASKT.BAS"
20 ' ATARI MICROSOFT BASIC II
```

```

30 GRAPHICS 0 : ' CLEAR SCREEN
40 POKE 82,0 : ' SET LEFT MARGIN TO 0
50 PRINT "PROGRAM: BASKT.BAS"
60 PRINT
70 PRINT "BOUNCES","DISTANCE"
80 PRINT
90 LET D=6
100 FOR X=1 TO 12
110 LET D=D+12.0*(2.0/3.0)^X
120 PRINT X,D
130 NEXT X
140 END

```



The screenshot shows the output of the BASIC program. It displays a table with two columns: 'BOUNCES' and 'DISTANCE'. The 'BOUNCES' column lists integers from 1 to 12. The 'DISTANCE' column shows the corresponding values calculated by the program, which are 14, 19.3333, 22.8889, 25.2593, 26.8395, 27.893, 28.5953, 29.0636, 29.3757, 29.5838, 29.7225, and 29.815. The program title 'PROGRAM: BASKT.BAS' is at the top, and a prompt character '>' is visible at the bottom left.

BOUNCES	DISTANCE
1	14
2	19.3333
3	22.8889
4	25.2593
5	26.8395
6	27.893
7	28.5953
8	29.0636
9	29.3757
10	29.5838
11	29.7225
12	29.815

It can be seen, by means of a contemporary example, how a progression can be used to solve a problem. It also solves a problem which would be quite difficult to do by hand, especially after the first few terms.

Chapter Two

Calculator

The solution to this problem illustrates how the computer can be used as a power calculator.

The problem is:

A boy on a bicycle and a man in a car start at the same time from town A for town B, 100 miles away. They travel over the same road at 6 and 40 miles per hour respectively. When the man in the car reaches town B, he will stop for 15 minutes and then start back again. How many hours will the boy on the bicycle have traveled when he meets the car on its return trip?

Let's set up our equations first. Remember $distance = rate * time$ or $T = \frac{D}{R}$. Letting T1 be the time the car has traveled before it starts the return trip, then:

$$T1 = \frac{110 \text{ mi}}{40 \text{ mph}} + \frac{1}{4} \text{ hr.}$$

The bicycle has traveled $T1 * 6\text{mph}$ at the time the car starts back. Hence the time until they meet is

$$T2 = \frac{110 - T1 * 6}{40 \text{ mph} + 6 \text{ mph}}$$

The total time is $T1 + T2$, so now we can easily write a program to solve the problem:

```
10 ' SAVE "D1:BICYCL1.BAS"
20 ' ATARI MICROSOFT BASIC II
30 GRAPHICS 0 : ' CLEAR SCREEN
40 POKE 82,0 : ' SET LEFT MARGIN TO 0
50 PRINT "PROGRAM: BICYCL1.BAS"
60 PRINT
70 LET T1=(110.0/40.0)+(1.0/4.0)
80 LET T2=(110.0-6.0*T1)/(40.0+6.0)
90 LET D = T1+T2
100 PRINT "CYCLIST HAS TRAVELED";D;"HOURS WHEN BOTH"
```

```
110 PRINT "MEET."  
120 END
```



Not very difficult and the numbers “come out even”. But they don’t have to. Let’s try it over a distance of 112 miles with speeds of 7.5 mph and 46 mph respectively. The computer solves this one just as easily. Can you?

```
70 LET T1=(112.0/46.0)+(1.0/4.0)  
80 LET T2=(112.0-7.5*T1)/(46.0+7.5)
```

The full program:

```
10 ' SAVE "D1:BICYCL2.BAS"  
20 ' ATARI MICROSOFT BASIC II  
30 GRAPHICS 0 : ' CLEAR SCREEN  
40 POKE 82,0 : ' SET LEFT MARGIN TO 0  
50 PRINT "PROGRAM: BICYCL2.BAS"  
60 PRINT  
70 LET T1=(112.0/46.0)+(1.0/4.0)  
80 LET T2=(112.0-7.5*T1)/(46.0+7.5)  
90 LET D = T1+T2  
100 PRINT "CYCLIST HAS TRAVELED";D;"HOURS WHEN BOTH"  
110 PRINT "MEET."  
120 END
```

```
PROGRAM: BICYCL2.BAS
CYCLIST HAS TRAVELED 4.40187
HOURS WHEN BOTH
MEET.
>
```

Computers can be used as powerful calculators too.

Chapter Three

Computer Assisted Instruction

This program demonstrates a simplified Computer Assisted Instruction drill and practice routine. The user determines how many digits each problem will contain. Then, ten addition problems are produced with the user providing the answer after each problem is presented. After the last problem is correctly answered, the score is printed with an appropriate comment.

```
10 ' SAVE "D1:CAIADD1.BAS"
20 ' ATARI MICROSOFT BASIC II
30 GRAPHICS 0 : ' CLEAR SCREEN
40 POKE 82,0 : ' SET LEFT MARGIN TO 0
50 RANDOMIZE
60 PRINT "PROGRAM: CAIADD1.BAS"
70 PRINT
80 PRINT "HOW MANY DIGITS";
90 INPUT N
100 LET W=0
110 LET P=2 : ' NUMBER OF PROBLEMS
120 FOR X=1 TO P
130 LET Q=0
140 LET A=INT(10^N*RND(0))
150 LET B=INT(10^N*RND(0))
160 PRINT
170 PRINT "    ";A
180 PRINT "  +";B
190 PRINT "  -----"
200 PRINT "    ";
210 INPUT G
220 PRINT
230 IF G=A+B THEN 270
```

```

240 GOSUB 360
250 PRINT "WHAT?? TRY AGAIN"
260 GOTO 160
270 PRINT "THAT'S RIGHT."
280 IF X<P THEN PRINT "TRY ANOTHER."
290 NEXT X
300 PRINT "YOU GOT";P-W;"CORRECT THE FIRST TIME"
310 IF W<=3 THEN 340
320 PRINT "BUT YOU MISSED";W
330 STOP
340 PRINT "GOOD WORK!"
350 GOTO 400
360 LET Q=Q+1
370 IF Q>1 THEN 250
380 LET W=W+1
390 RETURN
400 END

```

```

PROGRAM: CAIADD1.BAS
HOW MANY DIGITS? 2
  38
+ 83
----
? 121
THAT'S RIGHT.
TRY ANOTHER.
  25
+ 18
----
? 43
THAT'S RIGHT.
YOU GOT 2 CORRECT THE FIRST TIME
GOOD WORK!
>

```

This program could be used for drill and practice in a math class. However, it was designed to merely demonstrate the basic idea behind a traditional Computer Assisted Instruction program. It may be used to give some clues for writing more sophisticated CAI programs.

This next program is a somewhat more advanced version of CAI-ADD.

The extensions include questions on how many problems you wish to do and how many digits you wish to work with. The computer suggests working with 4 or fewer digits if the user specifies 5 or more (this limit can be changed in line 130).

Also, the columns of numbers are right justified in lines 250 and 260. This is a useful technique for formatting any kind of output. The generalized form of the statement is:

```
PRINT TAB(W-INT(LOG(N)/LOG(10)+1));N
```

“W” represents the width of the field while N is the number. You save a little computing time by using the numeric value of LOG(10), i.e., 2.302585 rather than computing the log every time.

Lines 440 to 570 replace line 270 in the CAI-ADD.BAS program. These statements simply print out one of four alternative messages of reinforcement when the correct answer is given.

```
10 ' SAVE "D1:CAIADD2.BAS"  
20 ' ATARI MICROSOFT BASIC II  
30 GRAPHICS 0 : ' CLEAR SCREEN  
40 POKE 82,0 : ' SET LEFT MARGIN TO 0  
50 RANDOMIZE  
60 PRINT "PROGRAM: CAIADD2.BAS"  
70 PRINT  
80 PRINT "HOW MANY PROBLEMS";  
90 INPUT P  
100 PRINT  
110 PRINT "HOW MANY DIGITS";  
120 INPUT N  
130 IF N<5 THEN 190  
140 PRINT  
150 PRINT "THAT'S TOO MANY FOR MY TASTE. WHY NOT"  
160 PRINT "TRY 4 OR FEWER DIGITS, ALL RIGHT?"  
170 PRINT "NOW THEN, HOW MANY DIGITS";  
180 GOTO 120  
190 LET W=0
```

```


200 X=X+1
210 Q=0
220 A=INT(10^N*RND(0))
230 B=INT(10^N*RND(0))
240 PRINT
250 PRINT TAB(6-INT(LOG(A)/2.302585+1));A
260 PRINT TAB(5-INT(LOG(B)/2.302585+1));"+";B
270 PRINT " ----"
280 PRINT " ";
290 INPUT G
300 PRINT
310 IF G=A+B THEN 430
320 IF Q<2 THEN 380
330 PRINT "THAT'S THE THIRD TIME YOU MISSED THAT
PROBLEM."
340 PRINT "THE CORRECT ANSWER IS ";A+B;". "
350 PRINT
360 PRINT "LET'S GO ON TO ANOTHER."
370 GOTO 200
380 Q=Q+1
390 IF Q>1 THEN 410
400 W=W+1
410 PRINT "WHAT? TRY AGAIN."
420 GOTO 240
430 IF X=P THEN 580
440 Y=Y+1
450 IF Y<>1 THEN 480
460 PRINT "VERY GOOD!"
470 GOTO 200
480 IF Y<>2 THEN 520
490 PRINT
500 PRINT "CORRECT. HERE'S ANOTHER."

```

```

510 GOTO 200
520 IF Y<>3 THEN 550
530 PRINT "YOU'RE A REAL MATH GIANT!"
540 GOTO 200
550 Y=0
560 PRINT "SUPER !!"
570 GOTO 200
580 PRINT
590 PRINT "YOU GOT";P-W;"CORRECT THE FIRST TIME"
600 IF W<=3 THEN 630
610 PRINT "BUT YOU MISSED";W
620 GOTO 640
630 PRINT "GOOD WORK!"
640 END

```



```

PROGRAM: CAIADD2.BAS
HOW MANY PROBLEMS? 2
HOW MANY DIGITS? 2
  +  1
+ 58
----
? 59
VERY GOOD!
  +  7
+ 36
----
? 43
YOU GOT 2 CORRECT THE FIRST TIME
GOOD WORK!
>

```

Chapter Four

Series Convergence

The CONVRG series of programs will calculate:

1. e by an infinite series.
2. π by inscribed and circumscribed polygon.
3. π by infinite series $(1 - 1/3 + 1/5 - 1/7 + 1/9...)$

Approach

The above methods are commonly taught to show how and may be accurately calculated; however, it is difficult to carry out the calculations very far using pencil and paper alone.

The portion of the program to converge e using the series

$$(1 + 1/1 + 1/2 + 1/6 + 1/24 + 1/120 + \dots)$$

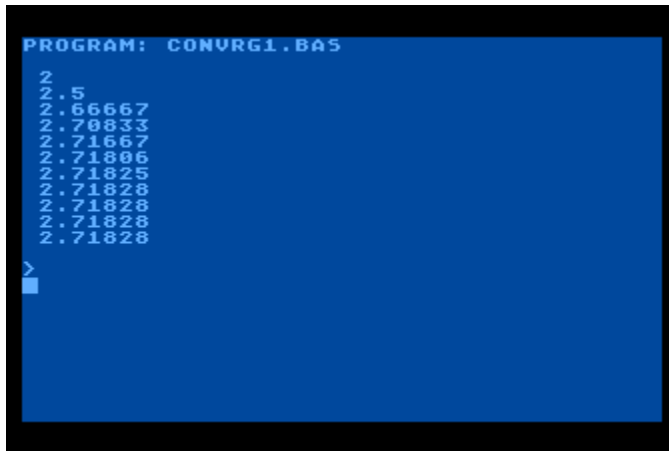
is as follows:

```
10 ' SAVE "D1:CONVRG1.BAS"
20 ' ATARI MICROSOFT BASIC II
30 GRAPHICS 0 : ' CLEAR SCREEN
40 POKE 82,0 : ' SET LEFT MARGIN TO 0
50 PRINT "PROGRAM: CONVRG1.BAS"
60 PRINT
70 LET R=0
80 LET E=1
90 LET I=I+1
100 LET D=1
110 FOR J=1 TO I
120 LET D=D*J
130 NEXT J
140 LET E=E+1/D
150 PRINT E
```

```

160 IF R = E THEN 190
170 R = E
180 GOTO 90
190 END

```



A screenshot of a BASIC program window titled "PROGRAM: CONVRG1.BAS". The window has a blue background and a black border. It displays a list of numerical values, likely representing the convergence of a sequence. The values are: 2, 2.5, 2.66667, 2.70833, 2.71667, 2.71806, 2.71825, 2.71828, 2.71828, 2.71828, and 2.71828. A cursor is visible at the bottom left of the window.

The portion of the program to converge on π by means of inscribed and circumscribed polygons is as follows:

```

10 ' SAVE "D1:CONVRG2.BAS"
20 ' ATARI MICROSOFT BASIC II
30 GRAPHICS 0 : ' CLEAR SCREEN
40 POKE 82,0 : ' SET LEFT MARGIN TO 0
50 PRINT "PROGRAM: CONVRG2.BAS"
60 PRINT
70 LET N=6
80 LET N=2*N
90 LET X=360/(N*57.29578)
100 LET R1=N*SIN(X)*COS(X)/2
110 LET R2=N*TAN(X)/2
120 PRINT R1,R2
130 IF R1=R2 THEN 150
140 GOTO 80
150 END

```

```
PROGRAM: CONVRG2.BAS
2.59808      3.4641
3.10583      3.21539
3.13263      3.15966
3.13935      3.14609
3.14103      3.14271
3.14141      3.14187
3.14141      3.14162
3.14146      3.14151
3.14139      3.14141
3.14121      3.14121
3.14083      3.14083
3.14006      3.14006
3.13853      3.13853
3.13547      3.13547
>
```

The third option, converging on π using an infinite series converges very slowly. Therefore, only every 500th value is displayed.

```
10 ' SAVE "D1:CONVRG3.BAS"
20 ' ATARI MICROSOFT BASIC II
30 GRAPHICS 0 : ' CLEAR SCREEN
40 POKE 82,0 : ' SET LEFT MARGIN TO 0
50 PRINT "PROGRAM: CONVRG3.BAS"
60 PRINT
70 LET C=0
80 LET S=1
90 LET I=1
100 LET Q=Q+1
110 LET P=P+S/I
120 LET I=I+2
130 LET S=-S
140 IF Q<499 THEN 100
150 LET Q=0
160 PRINT P*4
170 IF C>12 GOTO 200
180 LET C=C+1
190 GOTO 100
```


200 END

```
PROGRAM: CONVRG3.BAS
3.14364
3.1407
3.14243
3.14132
3.14228
3.1416
3.14228
3.1418
3.14233
3.14197
3.14241
3.14212
3.1425
3.14227
>
```

Chapter Five

Extended Calculator

This program illustrates how the computer, used as an extended calculator, can solve the following problems:

35 persons per 1000 have high blood pressure. 80% of those with high blood pressure drink, and 60% without high blood pressure drink. Estimate the fraction of drinkers with high blood pressure.

This problem requires the solution of several simple quotations. If we let H = the number of people with high blood pressure, then:

$$H = 35$$

And if $H1$ = the number of people with high blood pressure who drink, then:

$$H1 = 0.80 * H$$

Letting $L1$ = the number of people with low blood pressure who drink yields:

$$L1 = 0.60 * (1000 - H)$$

Now, we can solve for the number of drinkers, D :

$$D = H1 + L1$$

Finally, the percentage of drinkers with high blood pressure is:

$$X = H1 * 100/D$$

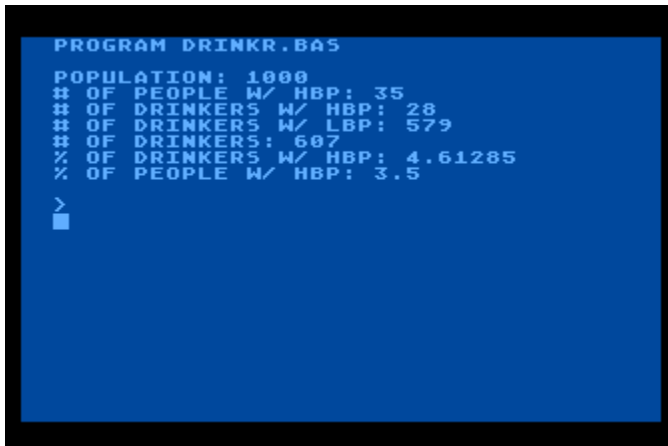
The program below solves the problem in a jiffy.

```
10 ' SAVE "D1:DRINKR.BAS"
20 ' ATARI MICROSOFT BASIC II
30 GRAPHICS 0 : ' CLEAR SCREEN
40 POKE 82,0 : ' SET LEFT MARGIN TO 0
50 PRINT "PROGRAM: DRINKR.BAS"
```

```

60 PRINT
70 LET P = 1000
80 LET H = 35
90 LET H1 = 0.8 * H
100 LET L1 = 0.6 * (P-H)
110 LET D=H1+L1
120 LET X=H1 * 100/D
130 LET B=H*100/1000
140 PRINT "POPULATION:";P
150 PRINT "# OF PEOPLE W/ HBP:";H
160 PRINT "# OF DRINKERS W/ HBP:";H1
170 PRINT "# OF DRINKERS W/ LBP:";L1
180 PRINT "# OF DRINKERS:";D
190 PRINT "% OF DRINKERS W/ HBP:";X
200 PRINT "% OF PEOPLE W/ HBP:";B
210 END

```



```

PROGRAM DRINKR.BAS
POPULATION: 1000
# OF PEOPLE W/ HBP: 35
# OF DRINKERS W/ HBP: 28
# OF DRINKERS W/ LBP: 579
# OF DRINKERS: 607
% OF DRINKERS W/ HBP: 4.61285
% OF PEOPLE W/ HBP: 3.5
>

```

This type of problem can be written directly in BASIC without going through the steps above. Recognizing this type of problem readily will save lots of pencil-pushing time.

Chapter Six

ROOTS

ROOTS finds all the roots of any function between -20 and 20. The function may be quadratic, cubic, trigonometric, or any combination.

The method used here involves evaluating the function at incremental values, finding places where the value of the function changes sign and then, by successive approximations, finding the zero point. The method is similar to the commonly taught "Newton's Method".

Before executing the program, enter the function to be evaluated as a DEF FNA(X) statement as line 70. For example:

```
70 DEF FNA(X) = 2*X^3+11*X^2-31*X-180
70 DEF FNA(X) = SIN(X)-.5
70 DEF FNA(X) = X-4
```

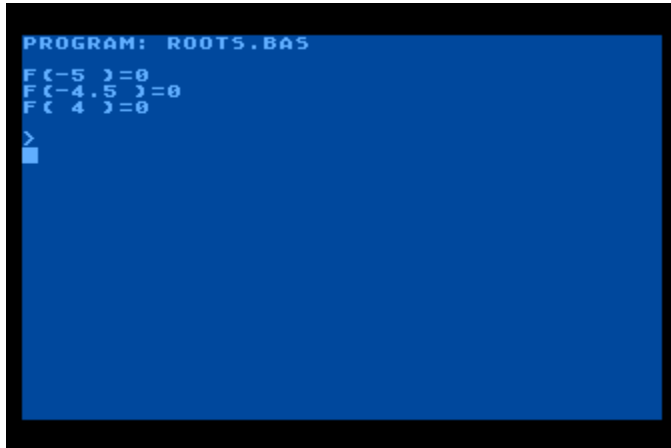
Here is the program. It takes a moment to run.

```
10 ' SAVE "D1:ROOTS.BAS"
20 ' ATARI MICROSOFT BASIC II
30 GRAPHICS 0 : ' CLEAR SCREEN
40 POKE 82,0 : ' SET LEFT MARGIN TO 0
50 PRINT "PROGRAM: ROOTS.BAS"
60 PRINT
70 DEF FNA(X)=2*X^3 + 11*X^2 - 31*X - 180
80 LET Z1=-200
90 FOR I=-19.9 TO 20
100 IF SGN(FNA(I))=SGN(FNA(I+1)) THEN 270
110 LET K=I
120 LET J=I+1
130 IF FNA(K)<FNA(J) THEN 170
```

```

140 LET Z=K
150 LET K=J
160 LET J=Z
170 LET Z=(K+J)/2
180 IF FNA(Z)< 0 THEN 210
190 LET J=Z
200 GOTO 220
210 LET K=Z
220 IF ABS(FNA(Z))>5.000000E-05 THEN 170
230 LET Z=SGN(Z)*INT(ABS(Z)*10000+.5)/10000
240 IF Z=Z1 THEN 270
250 PRINT "F("Z")=0"
260 LET Z1=Z
270 NEXT I
280 END

```



Change line 70 to:

```
70 DEF FNA(X) = SIN(X)-.5
```

And rerun the program

```
PROGRAM: ROOTS.BAS
F(-18.3259)=0
F(-16.2315)=0
F(-12.0428)=0
F(-9.9483)=0
F(-5.7596)=0
F(-3.6651)=0
F(-.5236)=0
F(2.6179)=0
F(6.8068)=0
F(8.9011)=0
F(13.09)=0
F(15.1844)=0
F(19.3732)=0
>
■
```

Change line 70 to:

70 DEF FNA(X) = X-4

and rerun the program:

```
PROGRAM: ROOTS.BAS
F( 4 )=0
>
■
```

Chapter Seven

SETS

This chapter covers a series of programs to determine the intersection of two sets of numbers.

Two sets of numbers can be combined to yield a third set by the operation of intersection. The intersection of two sets A and B is the set that contains all elements that belong to both A and B. It does not contain any other elements. The intersection is usually written as $A \cap B$.

For example, if $M = \{0, 2, 4, 6\}$ and $K = \{1, 2, 3, 4\}$, then $M \cap K = \{2, 4\}$.

The computer can be instructed to find the intersection of two sets. A program for finding the intersection of sets $x = \{1, 3, 5, \dots, 19\}$ and $y = \{2, 5, 8, \dots, 29\}$ is written below

```
10 ' SAVE "D1:SETS1.BAS"
20 ' ATARI MICROSOFT BASIC II
30 GRAPHICS 0 : ' CLEAR SCREEN
40 POKE 82,0 : ' SET LEFT MARGIN TO 0
50 PRINT "PROGRAM: SETS1.BAS"
60 PRINT
70 PRINT "THE INTERSECTION OF SETS X AND Y IS:"
80 FOR X=1 TO 19 STEP 2
90 FOR Y=2 TO 29 STEP 3
100 IF X=Y THEN 140
110 NEXT Y
120 NEXT X
130 GOTO 160
140 PRINT X
150 GOTO 120
160 END
```

```
PROGRAM: SETS1.BAS
THE INTERSECTION OF SETS X AND Y IS:
5
11
17
>
```

Using the computer, complicated sets which would be tedious to do by hand may be examined by defining one or both sets in a data statement. In the following example, set y is the same as above, but set x is given in the DATA statement.

```
10 ' SAVE "D1:SETS2.BAS"
20 ' ATARI MICROSOFT BASIC II
30 GRAPHICS 0 : ' CLEAR SCREEN
40 POKE 82,0 : ' SET LEFT MARGIN TO 0
50 PRINT "PROGRAM: SETS2.BAS"
60 PRINT
70 PRINT "THE INTERSECTION OF SETS X AND Y IS:"
80 READ X
90 IF X=-1 THEN 170
100 FOR Y=2 TO 29 STEP 3
110 IF X=Y THEN 140
120 NEXT Y
130 GOTO 80
140 PRINT X
150 GOTO 80
160 DATA 2,3,8,9,14,15,20,21,26,27,-1
170 END
```



```
PROGRAM: SETS2.BAS
THE INTERSECTION OF SETS X AND Y IS:
2
8
14
20
26
>
```

If there is a numerical pattern in the sets which intersect, there is also a pattern in the resultant intersecting set. In the first example above, for instance, set x steps 2 and set y steps by 3, hence the intersecting set steps by $2 * 3 = 6$. More complicated sets can be examined by computer than can easily be done by hand.

Chapter Eight

Simultaneous Equations

The programs in this chapter illustrate one way in which the computer can be used to solve relatively complex simultaneous equations.

The computer can be used to solve simultaneous equations by trial and error far faster than we humans can. For example, let's solve for x and y in the following equation:

$$2^x = \frac{16y}{3} \text{ and } 3^x = 27y$$

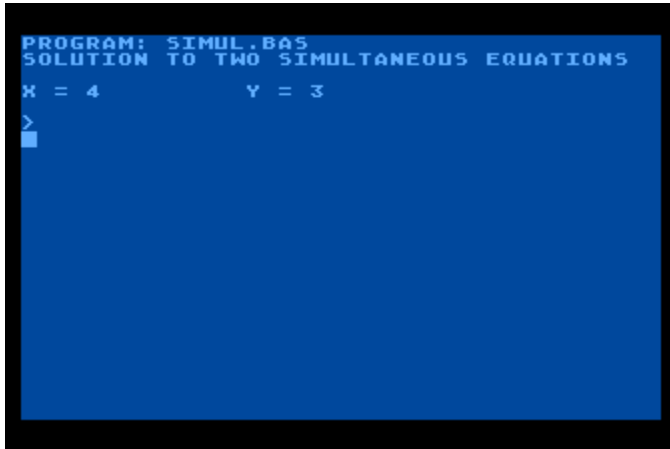
The program, which tries values of x and y between 0 and 100 until it reaches a solution or runs out of values is as follows:

```
10 ' SAVE "D1:SIMUL1.BAS"
20 ' ATARI MICROSOFT BASIC II
30 GRAPHICS 0 : ' CLEAR SCREEN
40 POKE 82,0 : ' SET LEFT MARGIN TO 0
50 PRINT "PROGRAM: SIMUL1.BAS"
60 PRINT "SOLUTION TO TWO SIMULTANEOUS EQUATIONS"
70 PRINT
80 FOR X=1 TO 100
90 FOR Y=1 TO 100
100 LET A=2^X
110 LET B=(16*Y)/3
120 LET C=3^X
130 LET D=27*Y
140 IF INT(A)<INT(B) THEN 200
150 IF INT(A)>INT(B) THEN 200
160 IF INT(C)<INT(D) THEN 200
170 IF INT(C)>INT(D) THEN 200
180 PRINT "X =";X,"Y =";Y
190 GOTO 230
```

```

200 NEXT Y
210 NEXT X
220 PRINT "NO INTEGER SOLUTION"
230 END

```



Let's try the same problem, but change the second equation slightly by changing the following lines:

```

80 FOR X=1 1 TO 10
90 FOR Y=1 TO 10
130 LET D=28*Y

```

Here is the modified program:

```

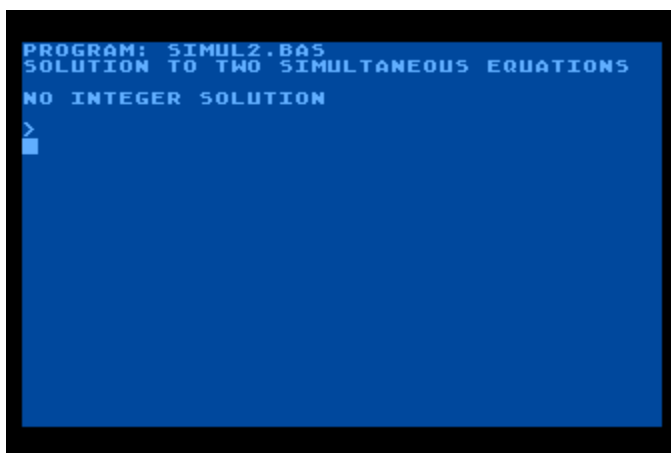
10 ' SAVE "D1:SIMUL2.BAS"
20 ' ATARI MICROSOFT BASIC II
30 GRAPHICS 0 : ' CLEAR SCREEN
40 POKE 82,0 : ' SET LEFT MARGIN TO 0
50 PRINT "PROGRAM: SIMUL2.BAS"
60 PRINT "SOLUTION TO TWO SIMULTANEOUS EQUATIONS"
70 PRINT
80 FOR X=1 1 TO 10
90 FOR Y=1 TO 10
100 LET A=2^X

```

```

110 LET B=(16*Y)/3
120 LET C=3^X
130 LET D=28*Y
140 IF INT(A)<INT(B) THEN 200
150 IF INT(A)>INT(B) THEN 200
160 IF INT(C)<INT(D) THEN 200
170 IF INT(C)>INT(D) THEN 200
180 PRINT "X =";X,"Y =";Y
190 GOTO 230
200 NEXT Y
210 NEXT X
220 PRINT "NO INTEGER SOLUTION"
230 END

```



Oh, oh. What went wrong?

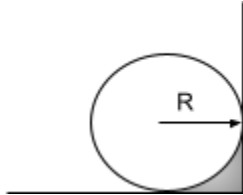
It can be seen that through trial and error, at least incrementing by whole numbers, does not always yield a solution, even trying 10,000 combinations. There must be another, better way.

Hint: Try combining the two equations and getting a solution for x using program ROOTS. Then y can be easily solved for.

Chapter Nine

AREA

The problem is to solve for the shaded area of the figure below for any value of the radius, R.



Start by remembering that the area of a circle and square are calculated as follows:

$$A_{\text{circle}} = \pi * R^2$$

$$A_{\text{square}} = SIDE^2 \text{ or } (2 * R)^2$$

The difference in area between a square and a circle inscribed within its borders is:

$$\Delta A = A_{\text{square}} - A_{\text{circle}}$$

and the area of one corner is:

$$A_{\text{corner}} = \Delta A / 4$$

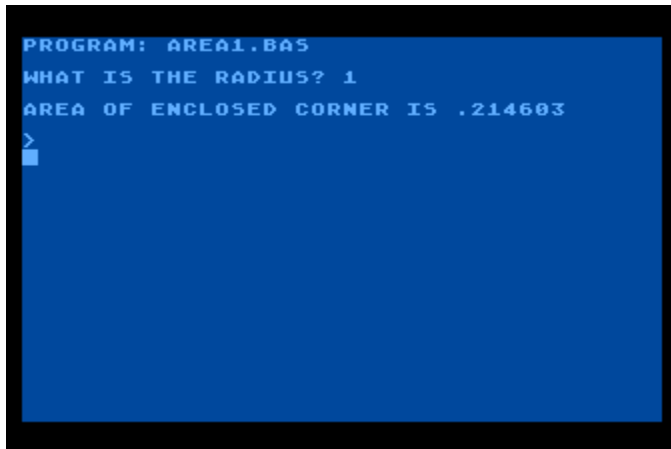
Now, let's write the program to perform this calculation for any input value of R.

By trying different values of R, the student can determine relationships in areas as a function of the components. Let's try several values of R and see what happens.

```
10 ' SAVE "D1:AREA1.BAS"
20 ' ATARI MICROSOFT BASIC II
30 GRAPHICS 0 : ' CLEAR SCREEN
40 POKE 82,0 : ' SET LEFT MARGIN TO 0
50 PRINT "PROGRAM: AREA1.BAS"
60 PRINT
```

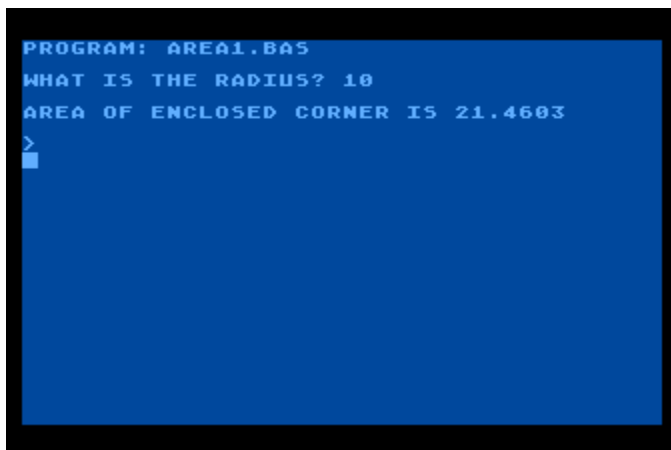
```
70 PRINT "WHAT IS THE RADIUS";
80 INPUT R
90 PRINT
100 LET P=3.14159
110 LET A1=P*R^2
120 LET A2=(2*R)^2
130 LET D=(A2-A1)/4
140 PRINT "AREA OF ENCLOSED CORNER IS";D
150 END
```

Run the program and enter 1 for the radius:

A screenshot of a BASIC program execution. The window has a blue background with white text. The text shows the program name 'PROGRAM: AREA1.BAS', the prompt 'WHAT IS THE RADIUS? 1', the calculated area 'AREA OF ENCLOSED CORNER IS .214603', and a prompt character '>' with a cursor.

```
PROGRAM: AREA1.BAS
WHAT IS THE RADIUS? 1
AREA OF ENCLOSED CORNER IS .214603
>
```

Rerun the program and enter 10 for the radius:

A screenshot of a BASIC program execution. The window has a blue background with white text. The text shows the program name 'PROGRAM: AREA1.BAS', the prompt 'WHAT IS THE RADIUS? 10', the calculated area 'AREA OF ENCLOSED CORNER IS 21.4603', and a prompt character '>' with a cursor.

```
PROGRAM: AREA1.BAS
WHAT IS THE RADIUS? 10
AREA OF ENCLOSED CORNER IS 21.4603
>
```

Rerun the program and enter 100 for the radius:

```
PROGRAM: AREA1.BAS  
WHAT IS THE RADIUS? 100  
AREA OF ENCLOSED CORNER IS 2146.03  
>  
■
```

Rerun the program and enter 2.158652 for the radius:

```
PROGRAM: AREA1.BAS  
WHAT IS THE RADIUS? 2.158652  
AREA OF ENCLOSED CORNER IS 1  
>  
■
```

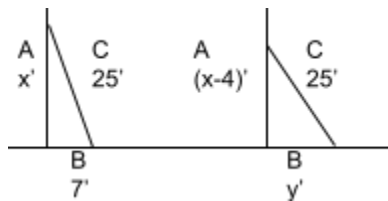
Chapter Ten

Pythagorean Theorem

The ladder program shows how the computer can be used to calculate the solution to the following word problem:

A ladder 25 feet long is placed so that its foot is 7 feet from the base of a building. When the top of the ladder slipped 4 feet down the side of the building, how far did the foot of the ladder slip?

The diagrams below show the two ladder positions.



By the Pythagorean theorem we know that $a^2 + b^2 = c^2$, hence, the equations needed to solve the above problem are:

$$x = \sqrt{25^2 - 7^2}$$

$$y = \sqrt{25^2 - (x - 4)^2}$$

and the amount of slippage of the base is:

$$z = y - 7$$

The program to do the problem is as follows:

```
10 ' SAVE "D1:LADDER1.BAS"
20 ' ATARI MICROSOFT BASIC II
30 GRAPHICS 0 : ' CLEAR SCREEN
40 POKE 82,0 : ' SET LEFT MARGIN TO 0
50 PRINT "PROGRAM: LADDER1.BAS"
60 PRINT
```



```

70 LET X=SQR(25*25-7*7)
80 LET Y=SQR(25*25-(X-4)*(X-4))
90 LET Z=Y-7
100 PRINT "LADDER BASE SLIPPED";Z;"FEET."
110 END

```



While the arithmetic in the problem above is not particularly “messy”, it still is no great joy to solve by hand. However, the computer would be equally happy to solve the problem if the ladder is 27.83 feet long and it was 7.62 feet from the wall originally. Would you?

Modify the following lines in the program:

```

70 LET X=SQR(27.83^2-7.62^2)
80 LET Y=SQR(27.83^2-(X-4)^2)
90 LET Z=Y-7.62

```

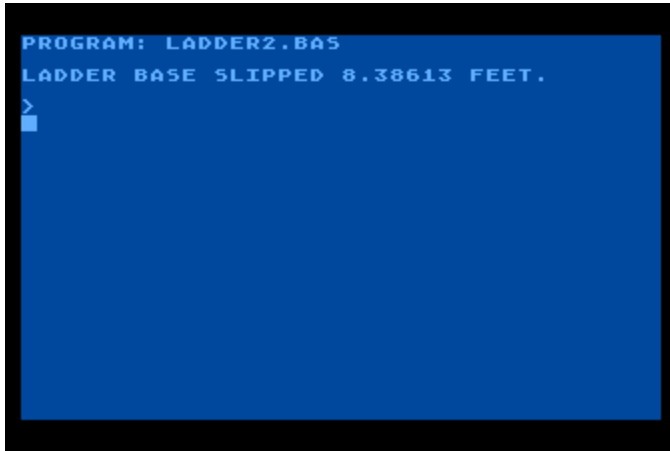
Here is the modified program:

```

10 ' SAVE "D1:LADDER2.BAS"
20 ' ATARI MICROSOFT BASIC II
30 GRAPHICS 0 : ' CLEAR SCREEN
40 POKE 82,0 : ' SET LEFT MARGIN TO 0
50 PRINT "PROGRAM: LADDER2.BAS"
60 PRINT
70 LET X=SQR(27.83^2-7.62^2)

```

```
80 LET Y=SQR(27.83^2-(X-4)^2)
90 LET Z=Y-7.62
100 PRINT "LADDER BASE SLIPPED";Z;"FEET."
110 END
```

A screenshot of a BASIC program execution. The window has a blue background with white text. The text displayed is: "PROGRAM: LADDER2.BAS", "LADDER BASE SLIPPED 8.38613 FEET.", and a prompt character ">" followed by a small white cursor block.

```
PROGRAM: LADDER2.BAS
LADDER BASE SLIPPED 8.38613 FEET.
>
```

Chapter Eleven

Guess

The technique of binary search has a number of mathematical and programming applications. In program GUESS.BAS, the computer chooses a random integer between 0 and 100 and the student must try to guess what it is in as few tries as possible.

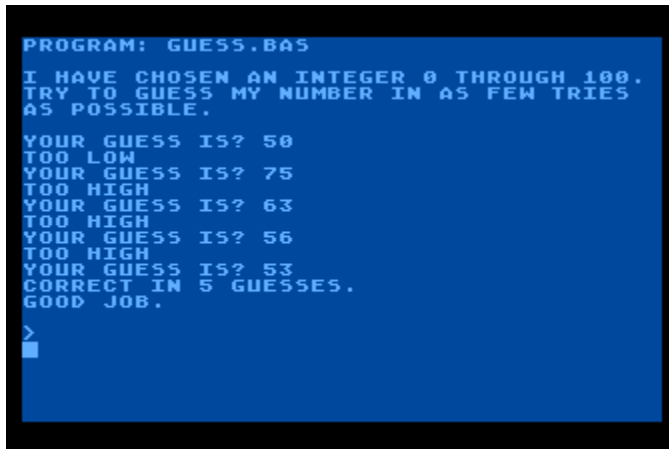
Note that the program does not reveal the technique of binary search, but merely alludes to the fact that no more than 7 guesses are necessary. The discovery of the idea of binary search will present itself after only a few runs of the program.

```
10 ' SAVE "D1:GUESS.BAS"
20 ' ATARI MICROSOFT BASIC II
30 GRAPHICS 0 : ' CLEAR SCREEN
40 POKE 82,0 : ' SET LEFT MARGIN TO 0
50 RANDOMIZE
60 PRINT "PROGRAM: GUESS.BAS"
70 PRINT
80 PRINT "I HAVE CHOSEN AN INTEGER 0 THROUGH 100."
90 PRINT "TRY TO GUESS MY NUMBER IN AS FEW TRIES"
100 PRINT "AS POSSIBLE."
110 PRINT
120 LET C=0
130 LET N=INT(RND(0)*100)
140 LET C=C+1
150 PRINT "YOUR GUESS IS";
160 INPUT G
170 IF N=G THEN 230
180 IF G>N THEN 210
190 PRINT "TOO LOW"
200 GOTO 140
210 PRINT "TOO HIGH"
```

```

220 GOTO 140
230 PRINT "CORRECT IN"C"GUESSES."
240 IF C>3 THEN 270
250 PRINT "YOU WERE LUCKY!"
260 STOP
270 IF C>7 THEN 300
280 PRINT "GOOD JOB."
290 GOTO 310
300 PRINT "BUT YOU SHOULDN'T NEED MORE THAN 7
GUESSES."
310 END

```



```

PROGRAM: GUESS.BAS
I HAVE CHOSEN AN INTEGER 0 THROUGH 100.
TRY TO GUESS MY NUMBER IN AS FEW TRIES
AS POSSIBLE.
YOUR GUESS IS? 50
TOO LOW
YOUR GUESS IS? 75
TOO HIGH
YOUR GUESS IS? 63
TOO HIGH
YOUR GUESS IS? 56
TOO HIGH
YOUR GUESS IS? 53
CORRECT IN 5 GUESSES.
GOOD JOB.
>

```

Try writing a program which reverses the role of the user and computer in GUESS.BAS. The computer should be programmed to guess a number the user has determined. After each guess, the user inputs 1, 2, or 3 for LOW, HIGH, or CORRECT and the program continues until the correct number is found.

The program should guess any number between 0 and 100 in 7 guesses provided all the user clues are consistent.

Gambling Simulation

This program is a simulation of a slot machine, or one-arm bandit, of the type popular at Las Vegas, Monte Carlo, and other gambling spas.

It is not a perfect simulation since three bars or three \$ are frequently, at the real casinos, a big jackpot. However, the operation of the slot machine as simulated by this program is probably more honest than most existing real machines. Calculate the average number of "pulls" needed to lose \$20.00. What is the probability of winning \$20.00?

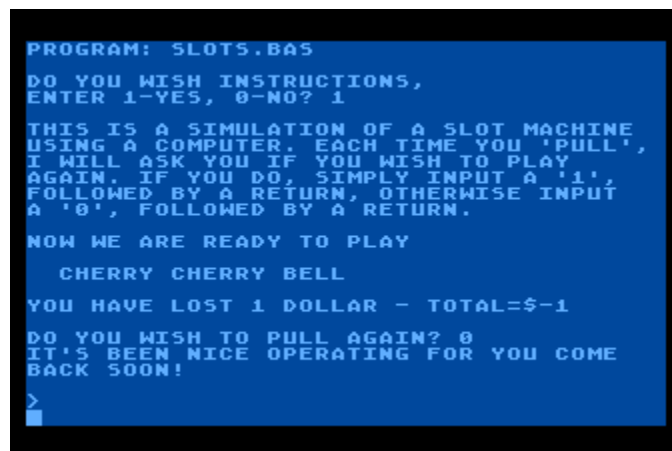
```
10 ' SAVE "D1:SLOTS.BAS"
20 ' ATARI MICROSOFT BASIC II
30 GRAPHICS 0 : ' CLEAR SCREEN
40 POKE 82,0 : ' SET LEFT MARGIN TO 0
50 RANDOMIZE
60 PRINT "PROGRAM: SLOTS.BAS"
70 PRINT
80 DIM D(3)
90 PRINT "DO YOU WISH INSTRUCTIONS,";
100 PRINT "ENTER 1-YES, 0-NO";
110 INPUT J
120 PRINT
130 IF J=0 THEN 220
140 PRINT "THIS IS A SIMULATION OF A SLOT MACHINE"
150 PRINT "USING A COMPUTER. EACH TIME YOU 'PULL',"
160 PRINT "I WILL ASK YOU IF YOU WISH TO PLAY"
170 PRINT "AGAIN. IF YOU DO, SIMPLY INPUT A '1',"
180 PRINT "FOLLOWED BY A RETURN, OTHERWISE INPUT"
190 PRINT "A '0', FOLLOWED BY A RETURN."
200 PRINT
210 PRINT "NOW WE ARE READY TO PLAY"
220 PRINT
```

```

230 PRINT TAB(2)" ";
240 FOR B1=1 TO 3
250 LET D(B1)=INT(RND(0)*6)+1
260 NEXT B1
270 FOR G1=1 TO 3
280 IF D(G1)=1 THEN 350
290 IF D(G1)=2 THEN 370
300 IF D(G1)=3 THEN 390
310 IF D(G1)=4 THEN 410
320 IF D(G1)=5 THEN 430
330 IF D(G1)=6 THEN 450
340 GOTO 660
350 PRINT "BELL  ";
360 GOTO 460
370 PRINT "BAR   ";
380 GOTO 460
390 PRINT "CHERRY ";
400 GOTO 460
410 PRINT "APPLE  ";
420 GOTO 460
430 PRINT "LEMON  ";
440 GOTO 460
450 PRINT "$      ";
460 NEXT G1
470 PRINT
480 IF D(1)<>D(2) THEN 500
490 IF D(2)=D(3) THEN 580
500 PRINT
510 IF D(1)=D(3) THEN 550
520 LET B=B-1
530 PRINT "YOU HAVE LOST 1 DOLLAR - TOTAL=$";B
540 GOTO 610

```

```
550 LET B=B+.50
560 PRINT "YOU HAVE WON 50 CENTS - TOTAL=$";B
570 GOTO 610
580 LET B=B+1
590 PRINT "YOU HAVE WON 1 DOLLAR - TOTAL=$";B
600 GOTO 610
610 PRINT TAB(20)"DO YOU WISH TO PULL AGAIN";
620 INPUT J
630 IF J=1 THEN 220
640 PRINT "IT'S BEEN NICE OPERATING FOR YOU COME"
650 PRINT"BACK SOON!"
660 END
```



```
PROGRAM: SLOTS.BAS
DO YOU WISH INSTRUCTIONS,
ENTER 1-YES, 0-NO? 1

THIS IS A SIMULATION OF A SLOT MACHINE
USING A COMPUTER. EACH TIME YOU 'PULL',
I WILL ASK YOU IF YOU WISH TO PLAY
AGAIN. IF YOU DO, SIMPLY INPUT A '1',
FOLLOWED BY A RETURN, OTHERWISE INPUT
A '0', FOLLOWED BY A RETURN.

NOW WE ARE READY TO PLAY

  CHERRY CHERRY BELL
YOU HAVE LOST 1 DOLLAR - TOTAL=$-1
DO YOU WISH TO PULL AGAIN? 0
IT'S BEEN NICE OPERATING FOR YOU COME
BACK SOON!
>
```

Chapter Thirteen

Patterns

This program prints an N by N matrix of diamond shapes, each one being R (odd number input) characters wide.

```
10 ' SAVE "D1:DIAMOND.BAS"
20 ' ATARI MICROSOFT BASIC II
30 GRAPHICS 0 : ' CLEAR SCREEN
40 POKE 82,0 : ' SET LEFT MARGIN TO 0
50 PRINT "PROGRAM: DIAMOND.BAS"
60 PRINT
70 PRINT "FOR A PRETTY DIAMOND PATTERN,"
80 PRINT "TYPE IN AN ODD NUMBER"
90 PRINT "BETWEEN 4 AND 10."
100 INPUT R
110 LET Q=INT(34/R)
120 FOR L=1 TO Q
130 LET X=1
140 LET Y=R
150 LET Z=2
160 FOR N=X TO Y STEP Z
170 PRINT TAB((2+R-N)/2);
180 FOR M=1 TO Q
190 LET C=1
200 FOR A=1 TO N
210 IF C=1 THEN 280
220 IF C=2 THEN 300
230 IF C=3 THEN 320
240 IF C=4 THEN 340
250 IF C=5 THEN 360
```



```
260 PRINT"!";
270 GOTO 390
280 PRINT "A";
290 GOTO 380
300 PRINT"T";
310 GOTO 380
320 PRINT"A";
330 GOTO 380
340 PRINT "R";
350 GOTO 380
360 PRINT "I";
370 GOTO 380
380 LET C=C+1
390 NEXT A
400 IF M=Q THEN 430
410 PRINT TAB(R*M+(2+R-N)/2);
420 NEXT M
430 PRINT
440 NEXT N
450 IF X<>1 THEN 500
460 LET X=R-2
470 LET Y=1
480 LET Z=-2
490 GOTO 160
500 NEXT L
510 END
```

```

PROGRAM: DIAMOND.BAS
FOR A PRETTY DIAMOND PATTERN,
TYPE IN AN ODD NUMBER
BETWEEN 4 AND 10.
? 5
  A      A      A      A      A      A
ATA ATA ATA ATA ATA ATA
ATARIATARIATARIATARIATARIATARI
ATA ATA ATA ATA ATA ATA
  A      A      A      A      A      A
  A      A      A      A      A      A
ATA ATA ATA ATA ATA ATA
ATARIATARIATARIATARIATARIATARI
ATA ATA ATA ATA ATA ATA
  A      A      A      A      A      A
  A      A      A      A      A      A
ATA ATA ATA ATA ATA ATA
ATARIATARIATARIATARIATARIATARI
ATA ATA ATA ATA ATA ATA
  A      A      A      A      A      A
ATA ATA ATA ATA ATA ATA

```

Chapter Fourteen

Plotting

This program accepts any input function, $f(x)$, at the time of program execution and plots that function from x_{\min} to m_{\max} in steps of S . The program also inserts the y axis, if it is crossed. Functions which can be plotted range from very simple to rather complex. All of the following are plottable functions:

Function	X Limits		Increment
$f(x) = 2 * x$	-15	+15	1
$f(x) = 30 * SIN(x)$	-5	+5	.25
$f(x) = x - x^2$	-5	+6	.5
$f(x) = 30 * SIN(.5 * x) * EXP(-.2 * ABS(x))$	0	+15	.25

```
10 ' SAVE "D1:PLOTFN.BAS"
20 ' ATARI MICROSOFT BASIC II
30 GRAPHICS 0 : ' CLEAR SCREEN
40 POKE 82,0 : ' SET LEFT MARGIN TO 0
50 PRINT "PROGRAM: PLOTFN.BAS"
60 PRINT
70 GOTO 150
80 PRINT "TO PLOT Y=F(X), THE USER MUST TYPE:"
90 PRINT
100 PRINT "100 GOTO 00"
110 PRINT "200 DEF FNA(X)=(THE EXPRESSION FOR F(X))"
120 PRINT "RUN"
130 PRINT
140 STOP
150 DEF FNA(X)=2*X
160 PRINT
```

```

170 PRINT "WHAT ARE THE INITIAL AND FINAL VALUES"
180 PRINT "OF X, AND THE INCREMENT SIZE";
190 INPUT X1,X2, S
200 PRINT
210 PRINT
220 PRINT TAB(33);"X =" ;X1
230 FOR X=X1 TO X2 STEP S
240 IF ABS(X)<.000001 THEN 290
250 LET Y=FNA(X)+35
260 IF Y>35 THEN 340
270 PRINT TAB(Y);"*";TAB(35);"."
280 GOTO 350
290 PRINT "-Y . . . . ."; : ' 26
300 PRINT " . . . . . +Y";TAB(FNA(X)+35);"*"
310 PRINT
TAB(4);"-30";TAB(14);"-20";TAB(24);"-10";TAB(35);"0";
320 PRINT TAB(44);" +10";TAB(54);" +20";TAB(64);" +30";
330 GOTO 350
340 PRINT TAB(35);".";TAB(Y);"*"
350 NEXT X
360 PRINT TAB(33);"X =" ;X2
370 PRINT
380 END

```

