

Probe 2000

1 Getting Started

Activity 1

After you turn on the computer and login, you will see several folders. Open the programming folder. Now, run QB45. (Be careful not to run two or more copies of the program, doing that will make the computer very slow.)

Type these lines on the computer, using your own name. Press the **<ENTER>** key after you type each line.

Type this program in.

```
PRINT "your name" <ENTER>
PRINT "MY NAME IS your name." <ENTER>
```

To run your program press Shift-F5.

The computer will print everything you type inside the quotation marks (called a **string**). If you make a typing mistake, use the back arrow key (<--), which is located above the **<ENTER>** to correct it. Now type these lines using your name and see what happens. Remember to press **<ENTER>** after each line.

```
PRINT "n a m e"
PRINT "  name"
PRINT "name"
PRINT "name  "
PRINT "na   me"
PRINT "namenamename"
```

Predict the **output** (what the computer prints on the screen) for each line below.

1. PRINT "COM PUTERS"
2. PRINT " ARE"
3. PRINT VERY "
4. PRINT "VERY FAST"
5. PRINT "M A C H I N E S."

Programming Tip: *If you forget the first quotation mark (as in line 3 above) or both quotation marks, the computer will print a zero. You don't need to type the end quotation mark if it is the last character in the line before you press <ENTER>. However, it is good programming practice to use both beginning and end quotation marks in all your PRINT statements.*

Write these lines as PRINT statements.

1. Today's date
2. Your name
3. Your street address
4. Your city
5. Your state and Zip Code
6. Your birth date
7. Your grade
8. Your school
9. Your favorite food
10. Your favorite sport

The computer remembers lines you number. In QBasic, however, it is not necessary to use line numbers. You may use letters if you wish to name lines as long as the letters are followed by a colon. Pressing *ALT, F, N* clears any previous programs from the

computer's memory. **END** shows the computer where to stop.

```
100 PRINT "I CAN"  
TwoHundred: PRINT "WRITE"  
30 PRINT "A PROGRAM."  
40 END
```

Now type SHIFT F5 to see your program. What happens when you type SHIFT F5 again?

Now write a program that prints the information about yourself that you filled in on lines 1-10 above. **CLS** moves the cursor to the top of the screen.

CLS

```
10 PRINT "TODAY'S DATE IS _____."  
  
PRINT "MY NAME IS _____."  
  
PRINT "I LIVE AT _____,"  
  
PRINT " _____,"  
  
PRINT " _____."  
  
PRINT "MY BIRTH DATE IS _____."  
  
PRINT "MY GRADE IS _____."  
  
PRINT "MY SCHOOL IS _____."  
  
PRINT "MY FAVORITE FOOD IS _____."  
  
PRINT "MY FAVORITE SPORT IS _____."  
  
END
```

Type SHIFT F5 to see your program. Some of your lines may have "wrapped around" the screen. There is room for 80 **characters** (letters, numbers, and symbols) and spaces across the screen.

Each program remains in the computer's memory until you type ALT, F N or turn off the

computer.

Now add this line to the program between line between the lin with your favorite sport and the **END** command.

GOTO 10

Run the program. What do you see on the screen?

To stop the program, hold down the *CTRL* key and press *BREAK* (located along the top right of the keyboard. Sometimes you may be required to press *SPACE* or **<ENTER>** also). The program now has a **loop** in it. The *GOTO* line tells the computer to go back to the line named 10 and repeat the program. It does this again and again until you tell it to stop. This kind of loop is called an *infinite loop*.

Programming Tip: When you press *CTRL-BREAK* to stop the running of a program, the computer tells you at which line the program was interrupted. If you press *F5* the computer will continue running the program from the point where it was stopped.

What happens if you make the program go back to a line other than line 10?

Add these *PRINT* statements between the lines of your program.

```
CLS
10 PRINT "TODAY'S DATE IS _____."
PRINT
PRINT "MY NAME IS _____."
PRINT
PRINT "I LIVE AT _____,"
PRINT
PRINT " _____,"
PRINT
PRINT " _____."
PRINT
PRINT "MY BIRTH DATE IS _____."
PRINT
PRINT "MY GRADE IS _____."
PRINT
PRINT "MY SCHOOL IS _____."
PRINT "MY FAVORITE FOOD IS _____."
```

```
PRINT  
PRINT "MY FAVORITE SPORT IS _____."  
GOTO 10  
END
```

Now run the program with the new lines. Use CTRL-BREAK to stop. What does the **PRINT** statement do when it is typed with nothing after it?

Programming Tip: *Line numbers or names are optional. They only need to be used when a GOTO command (or some similar command) is used.*

Activity 2

Programming Tip: To save time and typing, you can type a question mark (?) instead of the command **PRINT**. Experiment by typing ? "your name" on the computer. You will see that the computer lists the word **PRINT** wherever a question mark was typed (unless the question mark is contained inside the quotation marks of a string).

Type these lines. Use a question instead of typing out the word PRINT.

```
PRINT "A","B"  
PRINT "A","B","C"  
PRINT "A","B","C","D"  
PRINT "A","B",  
PRINT "A","B","C"
```

Now use your first name to run this program.

```
CLS  
10 PRINT "your name",  
GOTO 10  
END
```

Use CTRL-BREAK to stop the program. Change the comma at the end of the second line to a semicolon.

```
PRINT "your name";
```

Run the program. Write what each symbol tells the computer to do.

comma

semicolon

Programming Tip: It is not always necessary to use an **END** statement at the end of all programs. However, it is a requirement in some cases, so using it is a good habit to develop. In this book you will find an **END** statement at the end of all programs, even those with infinite loops, in which the program is stopped manually.

Predict the output of these programs, then run them on the computer. Type the lines exactly as they are shown.

```
CLS
PRINT "COMPUTERS ";
PRINT "ARE ";
PRINT "GREAT."
END
```

```
CLS
PRINT "BE CAREFUL . . ."
PRINT
PRINT "THIS IS","TRICKY."
END
```

Write a program that will print the following headings and information. Remember that each of the three print fields should have 7 or fewer characters and spaces in it.

FIRST NAME	LAST NAME	YEAR OF BIRTH
ABRAHAM	LINCOLN	1809
SUSAN B.	ANTHONY	1820
HARRIET	TUBMAN	1820

```
CLS

PRINT _____

PRINT _____

PRINT _____

PRINT _____

PRINT _____

PRINT _____

END
```

Run the program, then add some names and dates to the list.

Type this program on the computer.


```
10 PRINT "HI, MY NAME IS"  
20 PRINT  
30 PRINT "YOUR NAME"  
40 END
```

Now add these lines to the program between line 10 and 20 and line 20 and 30.

```
15 COLOR 4  
45 COLOR 2
```

The number following the **COLOR** command may be any number 1 to 16. (16 is black and cannot be seen.) The numbers 17 through 31 may be used to change the color of what is being printed and to make it blink at the same time.

Activity 3

You can use the TAB function to make characters appear in different places on the screen. Type this line on the computer.

```
PRINT TAB(75);"* * *"
```

The asterisks should be at the right side of the screen. Now run this program.

```
ALT, F N
CLS
PRINT TAB(5);"your name"
PRINT TAB(10);"your name"
PRINT TAB(15);"your name"
PRINT TAB (20);"your name"
END
```

TAB moves characters right to the column number you request in parentheses, much like the tab on a typewriter. There are 80 spaces for characters across the screen and 25 spaces down.

LOCATE does much the same thing as TAB. The first number in a locate statement is the vertical distance (distance down), the second number is the horizontal distance (distance right). Run this program that uses LOCATE.

```
ALT, F N
CLS
LOCATE 10, 16
PRINT "STARS"
LOCATE 13, 9
PRINT "**** S T A R S ****"
LOCATE 17, 15
PRINT "HURRAY!"
```

Programming Tip: To save typing, you can use a colon to separate two or three short statements on one line. Use this shortcut sparingly, however. Although statements are executed more quickly, lines overloaded with statements are harder to read when looking for errors, and are often harder to correct. Also, if a line has a GOTO statement in it, the GOTO statement must be the last statement in a line; any statements following a GOTO will be ignored.

Now add these lines after the end of you program, then run the program again.

```
COLOR 0, 7 : REM INVERSE
PRINT "*****"
COLOR 7, 0 : REM NORMAL
LOCATE 13, 15
PRINT "S T A R S"
COLOR 0, 7
LOCATE 13, 26
PRINT "*****"
COLOR 23, 0 : REM FLASH
LOCATE 17, 17
PRINT "WOW!"
COLOR 7, 0
END
```

Along with the colors listed above, there are also many others. Here is a table to show what each does:

The first number in the COLOR statement tells the computer what color to make the letters. The number after the comma tells the computer what color to make the screen behind the letters.

You may assign a color to certain parts of text. For more information on color, see chapter 5, activity 2.

0	Black
1	Blue
2	Green
3	Cyan
4	Red
5	Magenta
6	Brown
7	White
8	Gray
9	Blue
10	Green
11	Cyan
12	Red
13	Magenta

Write a program using any of the commands PRINT, GOTO, COLOR, TAB, and LOCATE.
You may also want to use the comma and semicolon.

END

Run your program on the computer.

2 Numbers

Activity 1

Use the PRINT command without quotation marks to tell the computer to perform math operations. Type these lines in and see the result.

```
PRINT 567
PRINT 56,789
PRINT 56789
PRINT -3.206
PRINT "5+3"
PRINT 5+3
PRINT 582.7+62
PRINT -5+9
PRINT 36.8-6.8
PRINT 8*7
PRINT 394*53
PRINT 42/6
PRINT 71/10
```

Write what each symbol tells the computer to do.

*

/

Why do you think the computer does not use an X or a dot to indicate multiplication?

The computer *truncates* (chops off) numbers to seven digits when it prints them. Type these lines.

```
PRINT 51/9
PRINT 62/11
PRINT 5/34
PRINT 1234567890
```

Write statements to calculate these expressions on the computer.

1. $18 - 9$
2. $28,100 + 4,575$
3. -5 times 3
4. 35 multiplied by itself
5. 98.38×30
6. $456 - 7$

Type these problems and try to figure out the order in which the computer performs the math operations.

PRINT 8+4-2	PRINT 4*2+5*3
PRINT 8-4+2	PRINT 56/7/2+3*3
PRINT 8-4-2	PRINT 3+12/2-5
PRINT 8*4+2	PRINT 8*(4+2)
PRINT 8+4*2	PRINT (8-4)/2
PRINT 8-4*2	PRINT 8-(4*2)
PRINT 8/4+2	PRINT 4*(2+5)*3
PRINT 8+4/2	PRINT 6*(2*(2+3))
PRINT 8*4/2	PRINT 2*(6+(8-4)*2/4)

Here are the computer's rules:

1. Operations in parentheses, if any, are done first, starting with the innermost set of parentheses.
2. Multiplication and division are done next, from left to right.
3. Then addition and subtraction are done, from left to right.

Predict the output of these PRINT lines, using the rules listed above.

- | | |
|--------------------------|----------------------|
| 1. PRINT $6+5*2$ ___ | 5. PRINT $4*3+(5-1)$ |
| 2. PRINT $(6+5)*2$ ___ | 6. PRINT $3+9/3-2$ |
| 3. PRINT $4*3+5-1$ ___ | 7. PRINT $(3+9)/3-2$ |
| 4. PRINT $4*(3+5)-1$ ___ | 8. PRINT $(3+9/3)-2$ |

Check your answers on the computer.

Make up some arithmetic expressions of your own. Predict the answers and check them on the computer.

- 1.
- 2.
- 3.
- 4.
- 5.
- 6.
- 7.

Activity 2

Type and run these programs on the computer.

```
CLS
PRINT "28 + 4 = ";28+4
PRINT "28 - 4 = ";28-4
PRINT "28 * 4 = ";28*4
PRINT "28 / 4 = ";28/4
END
```

Programming Tip: In the above program, spaces were inserted around the numbers in the string but not around the numbers in the numeric expression after the semicolon. It is a typing shortcut to eliminate the spaces in the expression, but good programming practice to insert them around numbers that will appear on the screen for ease of reading and clarity.

```
CLS
PRINT "14 DAYS = ";14 / 7;" WEEKS"
PRINT "35 DAYS = ";35 / 7;" WEEKS"
PRINT "309 DAYS = ";309 / 7" WEEKS"
END
```

```
CLS
PRINT "JOSÉ'S SCORE",392
PRINT "LAURA'S SCORE",375
PRINT "KIM'S SCORE",286
PRINT,"_"
PRINT
PRINT "TOTAL SCORE",392+375+286
PRINT
PRINT "AVERAGE SCORE",(392+375+286)/3
END
```



```
CLS
PRINT "NUMBER","SQUARE"
PRINT
PRINT 2, 2 * 2
PRINT 3, 3 * 3
PRINT 4, 4 * 4
PRINT 5, 5 * 5
END
```

Write a program that calculates and prints out how many seconds are in 1 day, 12 days, and 5,384 days.

Write a program that tells how much 324 doughnuts would cost if the price were \$1.25 per dozen.

Run your programs on the computer.

Activity 3

To perform calculations using exponents, press the SHIFT key and 6 to make the ^ symbol. Type these print lines on the computer and write the output.

8^2 PRINT 8 ^ 2

7^3 PRINT 7 ^ 3

5^4 PRINT 5 ^ 4

3^5 PRINT 3 ^ 5

The INT function prints the nearest whole integer for a number in parentheses (but not greater than the number in parentheses). Type these lines and write the output.

PRINT INT(38) ___ PRINT INT(-1.39)

PRINT INT(6.3248) ___ PRINT INT(-.0005)

PRINT INT(1.39) ___ PRINT INT(-.987)

PRINT INT(.347) ___ PRINT INT(3+4.9)

PRINT INT(.0005) ___ PRINT INT(3.3+4.9)

PRINT INT(.987) ___ PRINT INT(5/2)

PRINT INT(-38) ___ PRINT INT(1/3)

PRINT INT(-6.3248) ___ PRINT INT(4-1.5)

Use the computer to figure out if these statements are true or false.

1. $\text{INT}(3.6 + 7.29) = \text{INT}(3.6) + \text{INT}(7.29)$
2. $\text{INT}(3.6 + 7.42) = \text{INT}(3.6) + \text{INT}(7.42)$
3. for any X and Y, $\text{INT}(X+Y) = \text{INT}(X) + \text{INT}(Y)$
4. $4 * \text{INT}(3.24) = \text{INT}(4 * 3.24)$
5. $4 * \text{INT}(3.26) = \text{INT}(3.26)$

The SQR function prints the square root of a positive number in parenthesis. Type these

lines and write the output.

PRINT SQR(9)

PRINT SQR(100)

PRINT SQR(3.5)

PRINT SQR(.62)

PRINT SQR(2 * 6 + 26 / 2)

The ABS functions prints the absolute value (the distance it is from zero) of a number. Write the computer's output next to each line.

PRINT ABS(27)

PRINT ABS(.27)

PRINT ABS(0)

PRINT ABS(-.27)

PRINT ABS(5 * -5)

This program is a number guessing game. RND in line 50 stands for **random** and instructs the computer to choose a different answer each time you play the game. This is called a *random result* and can be compared to flipping a coin, rolling dice, turning a spinner, or dealing cards, and having no way of knowing what the result will be. You will learn about the LET, INPUT, and IF ... THEN statements in the next lesson. In line 1, **RANDOMIZE TIMER** means that the random numbers will all be completely randomly generated. **RANDOMIZE TIMER** should be used in all programs containing random number statements.

```
RANDOMIZE TIMER  
CLS  
PRINT "I HAVE A NUMBER BETWEEN 1 AND 100."  
PRINT "I WANT YOU TO GUESS IT."  
PRINT "IF YOU DON'T GUESS IT."  
PRINT "I WILL GIVE YOU A HINT."  
LET N = INT(RND*100)+1  
PRINT "YOUR GUESS";  
INPUT X  
IF X = N THEN GOTO 10  
PRINT "THE DIFFERENCE = ",ABS(N-X)  
PRINT "YOUR GUESS";  
INPUT X  
10 IF X = N THEN GOTO 20  
PRINT "I WIN, MY NUMBER = ";N:PRINT  
GOTO 30  
20 PRINT "YOU WIN!":PRINT  
30 PRINT "TO START AGAIN, TYPE SHIFT F5."  
END
```

Run the program. Try it out on a friend.

3 Variables

Activity 1

A **variable** is a name of a certain storage location in the computer's memory that can be assigned a value with a LET statement. In a LET statement, the equal sign means *stand for* or *replace*, rather than strictly *same as*. This value can be changed with another LET statement.

Predict and write the value of **numeric variable S** in the computer's memory for each line.

S = 4

S = 6 * 3

S = S + 2

S = S / 2 + S

PRINT S

END

Programming Tip: *Numeric variable names must begin with a letter. Try to make your variables descriptive, because it will make them easier to remember. Also, you cannot use certain **reserved words** as a variable name or within a variable name, or you will get a SYNTAX ERROR message. These words are commands and function names such as COLOR, ABS, LET, IF, and FOR.*

The value of a numeric variable stays the same until you give it a new value. What will be the value of each variable after each line is executed?

1. After $A = 10$ $A =$
2. After $B = 5$ $A = \underline{\quad}$ $B =$
3. After $C = A + B$ $A = \underline{\quad}$ $B = \underline{\quad}$ $C =$
4. After $B = 7$ $A = \underline{\quad}$ $B = \underline{\quad}$ $C =$
5. After $C = C - B$ $A = \underline{\quad}$ $B = \underline{\quad}$ $C =$
6. After $A = 2$ $A = \underline{\quad}$ $B = \underline{\quad}$ $C =$
7. After $D = C - A$ $A = \underline{\quad}$ $B = \underline{\quad}$ $C = \underline{\quad}$ $D =$
8. After $A = A - D$ $A = \underline{\quad}$ $B = \underline{\quad}$ $C = \underline{\quad}$ $D =$

Type and run these programs on the computer.

```
CLS
Y = 6
PRINT Y, Y + 2, Y - 2
PRINT Y * 2, Y / 2, Y ^ 2
END
```

Variables can be more than one letter long. In fact they can be up to 40 letters long and can contain numbers as well, but the first character must always be a letter.

```
CLS
Lbs = 3
Oz = Lbs * 16
PRINT Lbs; " POUNDS = "; Oz; " OUNCES"
END
```

```
CLS
FEET = 6
PRINT FEET * 12; " INCHES"
END
```

Programming Tip: Good programmers try to name variables in ways that help them to remember what the variable will store. That way, when programs get long they are not confused.

Type and run this program that uses GOTO to tell the computer to count.

```
CLS
N = 0
30 PRINT N
   N = N + 1
GOTO 30
END
```

Use CTRL-BREAK to stop the counting. To tell the computer to start counting at 50, change

N = 0 to N = 50

and run the program.

To count by even numbers, change

N = N + 1 to N = N + 2.


Now try the program.

You can use IF and THEN to tell the computer how far to count. Put this line right before the GOTO line.

IF N > 100 THEN END

To count backwards, use a minus sign. Make the following changes in your program.

```
N = N + 2
if N > 100 THEN END
```



```
N = N - 5
IF N < -100 THEN END
```

Programming Tip: When you tell the computer to end in an IF ... THEN statement, no separate END statement is required. It is good programming practice to have only one END statement per program.

Write a program to count by odd numbers from 1 to 11.

Write a program to count by fives from 50 to 500.

Write a program that will print the perimeter and the area of a rectangle that is 5 inches long and 2 inches wide. (Remember that $P = 2 \times L + 2 \times W$ and $A = L \times W$.)

Activity 2

The INPUT command lets you “talk” with the computer. It prints a question mark on the screen and tells the program to wait for you to type something. Type and run these programs that use INPUT.

```
CLS
20 PRINT "HOW MANY MILES"
INPUT Miles
PRINT
PRINT Miles; "MILES = ";
PRINT Miles * 5280; " FEET"
```

Add these lines to the end of your program and run it.

```
PRINT
GOTO 20
END
```

Use CTRL-BREAK to stop the program. Now type and run these programs.

```
CLS
20 PRINT "LENGTH IS";
INPUT L
PRINT "WIDTH IS";
INPUT W
PRINT "PERIMETER IS "; 2 * L + 2 * W
PRINT "AREA IS "; L * W
PRINT
GOTO 20
END
```

```

CLS
Again: PRINT "ENTER THREE SCORES,";
PRINT "SEPARATED BY COMMAS."
INPUT S1, S2, S3
PRINT
PRINT "TOTAL SCORE IS "; S1 + S2 + S3
PRINT "AVERAGE SCORE IS";
PRINT (S1 + S2 + S3) / 3
PRINT
GOTO Again
END

```

Suppose that newspapers cost carriers \$0.175 each and they sell them for \$.2 each. Write a program that asks the carriers how many papers they want and then prints the carrier cost and profit for that amount.

You can use a numeric variable to keep track of totals in a program. These are called *running* or *cumulative totals*. Type and run these programs on the computer.

```

CLS
TotalS = 0
30 PRINT "YOUR SCORE";
INPUT S
TotalS = TotalS + S
PRINT "TEAM SCORE IS NOW"; TotalS
PRINT
GOTO 30
END

```

```
CLS
TotalCost = 0
30 PRINT "HOW MANY ITEMS";
INPUT Items
PRINT "PRICE PER ITEM";
INPUT Price
Cost = Items * Price
TotalCost = TotalCost + Cost
PRINT "COST IS $"; Cost, "TOTAL IS $"; TotalCost
PRINT
GOTO 30
END
```

Try to eliminate unnecessary variables from programs. To eliminate variable C in the program above, you could do math inside the print statement by deleting **Cost = Items * Price** and changing the next two lines to:

```
TotalCost = TotalCost + Items * Price
PRINT "COST IS $"; Items * Price, "TOTAL IS $"; TotalCost
```

***Programming Tip:** To delete entire lines, move over them with the cursor while pressing the shift key. Then press the delete key.*

Write your own program that asks for the input of several numbers and keeps a cumulative total.

Run your program. Try it out on a friend.

You can use IF... THEN in an input program. Type and run these programs.

```
CLS
10 PRINT "HOW MANY INCHES";
INPUT I
IF I < 36 THEN PRINT I; " INCHES IS LESS THAN A YARD."
IF I = 36 THEN PRINT I; " INCHES EQUALS ONE YARD."
IF I > 36 THEN PRINT I; " INCHES IS MORE THAN A YARD."
GOTO 10
END
```

```
CLS
PRINT "* BINGO GAME *"
PRINT: PRINT "TYPE A NUMBER."
PRINT: INPUT N
IF N > 50 THEN GOTO TooSmall
IF N < 50 THEN GOTO TooLarge
IF N = 50 THEN GOTO GotIt
TooSmall: PRINT "TRY A SMALLER NUMBER.": GOTO 30
TooLarge: PRINT "TRY A LARGER NUMBER.": GOTO 30
GotIt: PRINT "BINGO!"
END
```

In line 60 below, <> means *not equal to*. Remember that a colon (:) lets you type another command on the same line.

***Programming Tip:** Use capital letters in the names you give to variables and line labels to help people tell words apart. Using capitals to do this is called "Humping". many programmers indent commands that are in a loop to make their programs more readable.*

```
CLS
PRINT "I'M THINKING OF A NUMBER"
PRINT "BETWEEN 1 AND 10."
40 PRINT "GUESS MY NUMBER."
    INPUT Q
    IF Q <> 7 THEN PRINT "TRY AGAIN.": GOTO 40
PRINT "YOU GUESSED IT!"
END
```

In Qbasic, you can use a second type of IF statement called a *Block IF* statement. You may put as lines as you like in a Block IF statement.

```
CLS
PRINT "I=7 THINKING OF A NUMBER"
PRINT "BETWEEN 1 AND 10"
PRINT "GUESS MY NUMBER"
INPUT Q
IF Q = 7 THEN
    PRINT "YOU GUESSED IT"
    PRINT "YOU WIN"
    END
ELSE
    PRINT "NO, TRY AGAIN"
END IF
END
```

The block if statement will keep going until it sees an END IF statement. An ELSE command can be put in to tell the computer what to do if the IF statement is not true.

See if you can change this program so the number to guess comes from an INPUT statement. Be sure to have the computer clear the screen using CLS before asking the person playing the game to start guessing.

Programming Tip: Good programmers make their programs easier to read using indentation and blank spaces. The statements inside an IF statement are usually indented.

In the program below, the variable C is a **counter**, which keeps track of how many times the loop has been repeated. Because line 60 tells the program to end, a separate END statement is not needed.

```
CLS
C = 0
LoopAgain: PRINT "THIS IS A"
            PRINT "GOTO LOOP."
            C = C + 1
60  IF C > 10 THEN END
GOTO LoopAgain
```

You can type PRINT C at the end of this program to see the value of C after the program has been run. You can make this program loop 20 times. Change the counter C in line 60 to this, run the program, then type PRINT C.

```
60 IF C > 20 THEN END
```

Type and run this program on the computer.

```
CLS
PRINT "* MULTIPLE CHOICE *"
PRINT "CHOOSE SOMETHING THAT SWIMS"
PRINT
PRINT " 1) POLEMONIUM"
PRINT " 2) ERITHACUS RUBECOLA"
PRINT " 3) CETACEA"
10 INPUT A
IF A = 1 THEN
    PRINT "SORRY, BUT THAT'S A FLOWER"
ELSE IF A = 2 THEN
    PRINT "SORRY, BUT THAT'S A ROBIN."
ELSE IF A = 3 THEN
    PRINT "CONGRATULATIONS, THAT'S A WHALE!"
ELSE
    PRINT "PLEASE TYPE 1, 2 OR A 3."
    GOTO 10
END IF
END
```

Notice that in this program, if the user types in any number except 1, 2 or 3 the computer will ask them to retype their answer.

Programming Tip: *It is a good idea to make your programs easy to use. Try to let the person running the program fix any typing mistakes they make by having the computer re-ask questions when what they typed doesn't make sense. Programs that do this are called Robust.*

You can simplify this program by using the SELECT CASE command.

```
CLS
PRINT "* MULTIPLE CHOICE *"
PRINT "CHOOSE SOMETHING THAT SWIMS"
PRINT
PRINT " 1) POLEMONIUM"
PRINT " 2) ERITHACUS RUBECOLA"
PRINT " 3) CETACEA"
10 INPUT A
SELECT CASE A
    CASE 1
        PRINT "SORRY, BUT THAT'S A FLOWER"
    CASE 2
        PRINT "SORRY, BUT THAT'S A ROBIN."
    CASE 3
        PRINT "CONGRATULATIONS, THAT'S A WHALE!"
    CASE ELSE
        PRINT "YOU NEED TO TYPE 1, 2, OR 3."
        GOTO 10
END SELECT
END
```

Try this program out on a friend.

The SELECT CASE command has some really nice options. For example:

CASE Command	When it is true
CASE 1, 3, 5, 7	For numbers 1, 3, 5, 7
CASE 1 TO 100	For numbers 1 to 100
CASE 1, 2, Var1, Var2	For 1 and 2 and what ever Var1 and Var2 happen to equal.
CASE IS < 5	For numbers less than 5
CASE IS > 200	For numbers greater than 200
CASE "A" TO "Z"	For any letter A through Z

Activity 3

You know that GOTO can make a loop in a program. The words FOR and NEXT can also make a loop. The computer executes the statements between the FOR and NEXT statements as many times as the FOR statement allows. Here is another way to count to 20 on the computer. Compare it to the counting programs that use LET and GOTO in Activity 1.

```
CLS  
10 FOR X = 0 TO 20  
    PRINT X  
NEXT X  
END
```

Run the program. Change line 10 to count by twos.

```
10 FOR X = 0 TO 20 STEP 2
```

The STEP command tells the computer to count by twos. Run the program. Change line 10 to this.

```
10 FOR X = 0 TO 20 STEP 3.25
```

Run the program then change line 10 to count to 20 by fives.

```
10
```

Count by threes to 60.

```
10
```

Change line 10 to this and try it.

```
10 FOR X = 100 TO 5 STEP -5
```

Can you count by twos backwards from 30 to -30.

```
10
```

This program has three FOR ... NEXT loops with three different variables: A, B, and C. Each loop is executed 100 times.

```
CLS
PRINT "* CONTINUING MESSAGE *"
FOR A = 1 TO 3000
    PRINT "YOU "
NEXT A
FOR B = 1 TO 3000
    PRINT "ARE";
NEXT B
FOR C = 1 TO 3000
    PRINT "W O N D E R F U L !"
NEXT C
END
```

This program counts as far as you tell it.

```
CLS
PRINT "COUNT HOW FAR";
INPUT A
FOR X = 1 TO A
    PRINT X
NEXT X
END
```

The next program asks you to tell it where to start counting and where to stop.

```
ALT, F N
CLS
10 PRINT "BEGINNING NUMBER";
INPUT B
PRINT "ENDING NUMBER";
INPUT E
FOR N = B TO E
    PRINT N
NEXT N
GOTO 10
END
```

Press CTRL BREAK to end this program.

A **nested loop** is a FOR ... NEXT loop inside another FOR ... NEXT loop. In the program below the loop with the variable K is inside the loop with the variable N.

```
CLS
10  FOR N = 1 TO 20
      FOR K = 1 TO N
            PRINT "*";
      NEXT K
      PRINT
    NEXT N
GOTO 10
END
```

This program has two loops nested inside a larger loop.

```
CLS
10  FOR L = 1 TO 10
      FOR X = 1 TO L
            PRINT "X";
      NEXT X
      PRINT "";
      FOR Y = 1 TO 11- L
            PRINT "O";
      NEXT Y
      PRINT
    NEXT L
GOTO 10
END
```

***Programming Tip:** FOR ... NEXT loops may be nested, but may not cross. It is a good idea to draw arrows, if only imaginary ones, to make sure no lines cross. Below is an example of loops that are not **nested** properly:*

```
CLS
10 FOR N = 10 TO 20
20  PRINT N
30  FOR I = 30 TO 40
40    PRINT I
50  NEXT N
60 NEXT I
70 END
```

² Try to find the problem in this

To uncross the loops, switch lines 50 and 60.

Activity 4

Type and run this program to print 20 integers chosen at random from the values 0 to 9.

```
CLS
FOR N = 1 TO 20
    Z = INT(RND * 10)
    PRINT Z
NEXT N
END
```

Line 10 tells the computer to print 20 numbers. Line 20 says those numbers should be 10 random integers, starting with zero, or 0 to 9. Change line 20 to this.

```
20 Z = INT(RND * 11) + 1
```

Run the program. Line 20 now tells the computer to print 11 random integers, starting with 1, or 1 to 11. The +1 tells the computer at which number to start. In the first program above, +0 was understood. Change line 20 to each line below, run the program, and write what random integers are generated.

```
20 Z = INT(RND * 5)
```

```
20 Z = INT(RND * 15) + 10
```

```
20 Z = INT(RND * 4) - 3
```

Write line 20 to generate random integers:

1. from 23 through 45
2. from -13 through 0
3. from -5 through -19

Give the lowest and highest possible values of X.

1. $X = \text{INT}(\text{RND} * 35) + 2$

2. $X = \text{INT}(\text{RND} * 10) - 43$

3. $X = \text{INT}(\text{RND} * 59) - 15$

4. $X = \text{INT}(\text{RND} * 13) - 97$

5. $X = \text{INT}(\text{RND} * 21) - 10$

Type this program, which is a simulation of the random results of throwing dice.

```
CLS
X = INT(RND * 6) + 1
Y = INT(RND * 6) + 1
PRINT "DIE 1 = "; X
PRINT "DIE 2 = "; Y
PRINT "TOTAL = "; X + Y
END
```

Run the program several times to see the random results.

This program is a simulation of tossing a coin. The computer will print twos to represent heads and ones to represent tails.

```
CLS
PRINT "NUMBER OF HEADS/TAILS"
PRINT "RANDOMLY GENERATED"
PRINT "FROM 100 COIN TOSSES."
Tails = 0: Heads = 0
FOR I = 1 TO 100
    Coin = INT(RND * 2) + 1
    PRINT Coin
    IF Coin = 1 THEN Tails = Tails + 1: GOTO 110
    Heads = Heads + 1
110 NEXT I
PRINT "HEADS = "; Heads
PRINT "TAILS = "; Tails
```

***Programming tip:** You can type two commands on a line if you put a colon between them. If you put two or more commands after an IF, they all run only when the IF is true.*

This program asks the user to supply a missing number in a subtraction problem.

```
CLS
PRINT "MISSING NUMBER QUIZ"
PRINT
PRINT "WHAT NUMBER IS MISSING?"
A = INT(RND * 11) + 1
B = INT(RND * 20) + 10
PRINT B; " - ? = "; A
TypeInAnswer: INPUT Missing
IF B = A + Missing THEN GOTO Right
PRINT "TRY AGAIN.": GOTO TypeInAnswer
Right: PRINT "EXACTLY RIGHT!"
END
```

Run the programs. Write a program to ask random multiplication questions. Make it as fancy as you can.

Activity 5

You know that characters and spaces inside a set of quotation marks are called a string. A variable that stands for a string of letters, numbers, symbols, and spaces is called a **string variable**. It looks the same as a numeric variable except that it has a dollar sign (\$) in the label. Run this program that shows what the word *BASIC* (the name of the language in which you are programming) means.

```
CLS
B$ = "BEGINNER'S": PRINT B$
A$ = "ALL-PURPOSE": PRINT A$
S$ = "SYMBOLIC": PRINT S$
I$ = "INSTRUCTION": PRINT I$
C$ = "CODE": PRINT C$
END
```

LEN stands for *length* and finds the number of characters (including spaces) in a string variable. Type and run this program on the computer.

```
CLS
PRINT "TYPE ANY WORD."
INPUT A$
PRINT "THE WORD HAS "; LEN(A$);
PRINT "LETTERS IN IT."
END
```

Each time you run the program the computer will tell you the number of letters of letters in the word you type as input.

LEFT\$(A\$,1) will tell you the first letter of a string variable. The program below shows how **LEN** and **LEFT\$** can be used.

```
CLS
PRINT "HOW MANY SPELLING WORDS DO YOU WANT"
PRINT "TO PRACTICE?"
PRINT "TYPE A NUMBER FROM 1 TO 10."
INPUT N
FOR I = 1 TO N
    CLS
    PRINT "TYPE A WORD."
    INPUT A$(I)
    PRINT "TYPE A WORD THAT RHYMES WITH YOUR WORD."
    INPUT B$(I)
```

```

NEXT I
PRINT "NOW LET'S PRACTICE SPELLING."
PRINT " *****"
PRINT
FOR X = 1 TO N
180 PRINT "THE WORD BEGINS WITH "; LEFT$(A$(X), 1)."
    PRINT
    PRINT "IT HAS"; LEN(A$(X); " LETTERS."
    PRINT
    PRINT "IT RHYMES WITH "; B$(X)."
    PRINT
    PRINT "TYPE THE WORD."
    INPUT C$:PRINT:PRINT
    IF C$<>A$(X) THEN PRINT "TRY AGAIN.":GOTO 180
    PRINT "RIGHT!"
NEXT X
END

```

Try this program out on a friend.

In A\$(X) the variable X is called a *subscript*. If there is more than one string assigned to a variable like A\$, it tells the computer which one you want to use in each statement. A\$(1) would be the first one. A\$(2) would be the second, and so on. When the *subscript* itself is a variable, like X in our example, as the subscript changes the computer uses different strings stored in A\$.

LEFT\$(A\$,2) will tell you the first two letters of a string variable, and so on. RIGHT\$ does much the same thing as LEFT\$. Type and run this program, using your first and last names.

```

CLS
LET A$ = "your name"
FOR N = 1 TO LEN(A$)
    PRINT LEFT$(A$, N)
NEXT N
END

```


4 Graphics

Activity 1

In QBasic, there are many ways of creating graphics programs. In this lesson, we are going to use screen 12, *High Resolution* (clarity) VGA graphics. Using the SCREEN 12 command makes the monitor show 640 *pixels* (dots) horizontally (across) by 480 vertically (down). Try the following program.

```
SCREEN 12  
COLOR 1  
LINE (0,0)-(640,0)  
LINE (640,0)-(640,480)  
LINE (640,480)-(0,480)  
LINE (0,480)-(0,0)
```

What does this program do?

The screen is set up as a grid. Along the top of the screen, 0 is the left of the screen, and 640 is the right. To the side, 0 is the top, and 480 is the bottom. So, a dot in the middle of the screen would be LINE (320,240). When expressed as two points connected by a hyphen, such as LINE (10,15)-(300,400), simply means draw a line from location 10,15 to location 300,400.

Write a program to draw a line from the top left corner of the screen to the bottom right corner of the screen. Remember to set the screen to 12 and put in a COLOR command with a number between 1 and 15.

Now write a program to write the first letter of your name.

Activity 2

Run this program to see all the colors SCREEN 12 can make.

```
SCREEN 12
FOR X = 1 TO 15
  COLOR X
  LINE (100,X)-(200,X)
NEXT X
```

You will notice that there are several different colors available to you. Run this program to see each of the colors and their numbers.

```
SCREEN 12
Ask: PRINT "WHAT COLOR NUMBER (0-15)->";
      INPUT C
      COLOR C
      LINE (100,100)-(300,300)
GOTO Ask
```

Here is a list of the colors and their numbers for SCREEN 12:

0 = Black	8 = Dark Grey
1 = Dark Blue	9 = High Dark Blue
2 = Green	10 = High Green
3 = Light Blue	11 = High Light Blue
4 = Red	12 = High Red
5 = Purple	13 = High Purple
6 = Brown	14 = Yellow
7 = Light Grey	15 = White

Run this program to draw a truck.

```
SCREEN 12
COLOR 3
LINE (100,100)-(400,100)
LINE (400,100)-(400,300)
LINE (100,100)-(100,300)
LINE (400,300)-(100,300)
LINE (400,200)-(470,200)
LINE (470,200)-(470,300)
LINE (400,300)-(470,300)
LINE (100,300)-(120,330)
LINE (120,330)-(150,330)
LINE (150,300)-(150,330)
LINE (300,300)-(300,330)
```

```
LINE (300,330)-(330,330)
LINE (330,300)-(330,330)
```

Boxes: In the previous programs, you used the line command to draw pictures and letters. If you used the line command in this way, it would take four separate line commands to draw a box. Fortunately, there is a much easier way. There is a special option in the line command just for drawing boxes. The command **LINE (100,100)-(300,300),,B** draws a box with an upper left hand corner at location 100,100 and a lower right hand corner at location 300,300. The place between the two commas is the place you could put the color number, if you wanted the box to be a different color from the drawings you made earlier.

Try this program to see how easy it is to make boxes:

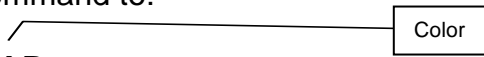
```
SCREEN 12
COLOR 4
LINE (20,20)-(50,50),,B
LINE (60,60)-(200,100),,B
LINE (10,10)-(500,400),,B
END
```

Try this program to see boxes in color:

```
SCREEN 12
FOR X= 1 TO 15
    COLOR X
    LINE (1,1)-(X*10,X*10),,B
NEXT X
END
```

The color changed for each box and the bottom corner moved down at an angle for each box. This program could be made one line shorter by taking out the COLOR command and changing the line command to:

```
LINE (1,1)-(X*10,X*10),X,B
```



Filling Boxes: QBasic has an option to fill boxes with color. If you place the letters **BF** at the end of a line command, QBasic will draw a box and fill it up.

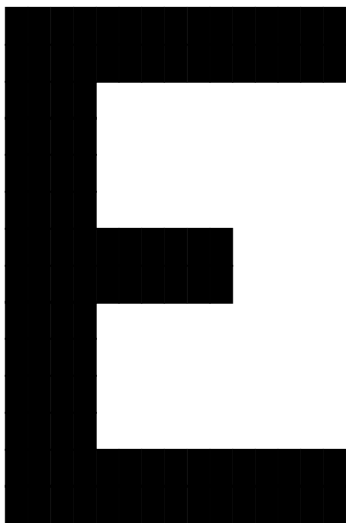
Type this program in and try it out.

```
SCREEN 12  
LINE (1,1)-(50,50),3,BF  
LINE (30,30)-(100,100),5,BF  
LINE (70,70)-(200,200),7,BF  
LINE (180,180)-(350,350),9,BF  
END
```

Notice how each box "covers over" the box draw before it.

Programming Tip: *Objects drawn first tend to look like they are in the background or behind everything else. Recently drawn objects look like they are in front or on top of earlier pictures. Good programmers handle the background of their graphics work first. They then program objects to appear on top of the backgrounds they created earlier.*

Write a program to draw the following shape:



Now Write a program that draws a house.

Activity 3

There are several other commands that will help you if you want to write more advance programs.

CIRCLE, not surprisingly, is a command that gives the ability to the user to draw a circle.

Here is a program to draw two circles on the screen.

```
SCREEN 12: COLOR 3
CIRCLE (200,200), 100
CIRCLE (100,100), 50
```

The location 200,200 is the center of a circle. 100 is the *radius* (how far out from the center to draw the circle.)

Type in and run this program.

```
SCREEN 12
COLOR 9
LINE (100,100)-(200,200),,B
COLOR 13
LINE (300,300)-(500,300)
LINE (500,300)-(250,200)
LINE (250,200)-(300,300)
COLOR 12
CIRCLE (100,300),60
```

As you can see, a series of shapes appears on the screen. Now add these lines to the end of your program.

```
PAINT (150,150),10,9
PAINT (350,250),11,13
PAINT (100,300),14,12
```

In line 130, PAINT (150,150) tells the computer to paint all the *pixels* on the screen around location 150,150 until another line is encountered. The number 10 is the color of the area is painted, and 9 tells the computer to stop painting when it reaches any boundaries of that color.

Activity 4

Run this program to make an object move across the screen.

```
SCREEN 12
FOR X = 1 TO 500 STEP .01
COLOR 3
LINE (X,200)-(X+100,200)
LINE (X,200)-(X,400)
LINE (X,300)-(X+100,300)
LINE (X+100,200)-(X+100,300)
COLOR 0
LINE (X,200)-(X+100,200)
LINE (X,200)-(X,400)
LINE (X,300)-(X+100,300)
LINE (X+100,200)-(X+100,300)
NEXT X
```

Save this program on your diskette. Try to make the object larger, change the color. How does your computer make the object seem to move? (Hint: look at the color chart for color number 0)

Programs that make things move are called *animation* programs. Run your animation program. Think of ways to modify it. Some ideas for programs might be a car that drives across the screen, a dog that runs, a flower that grows.

Another way to put animation in programs is to photograph sections of the screen in large variables called *arrays* and then make them appear in different places on the screen.

Type in this program and try it out.

```
DIM A(100)  
SCREEN 12  
CIRCLE (10,100),8  
CIRCLE (7,100),6  
CIRCLE(4,100),3  
GET (1,90)-(10,110),A  
X = 1  
DO WHILE X < 600  
    PUT (X,90), A, XOR  
    X = X + 1  
    PUT (X,90), A, XOR  
    FOR DELAY = 1 TO 10000: NEXT DELAY  
LOOP
```

The **GET** command "takes a picture" of a section of the screen and stores it in a variable named **A**. The **PUT** command takes the picture out of variable **A** and puts it on the screen so its left corner begins at the spot the numbers in the parentheses indicate.

One great advantage to programming animation in this way is that backgrounds are left intact. As the object passes across the screen, the **XOR** command in the **PUT** statement puts the background back in place.

Add these lines to your program after line 50 to see the program handle background.

```
52 LINE (100,1)-(150,400),,B  
54 LINE (120,1)-(200,400),,B
```

Change this program so that a flower travels across the screen.

5 Additional Features

Activity 1

Many of the programs you have seen thus far are short routines that could be used as a **subroutine**, or part of a larger program. **REM** stands for *remark* and is a note to yourself in the program. Use **REM** to label each part of a larger program so that when you look at your program you will see where each section begins and what it does. In Qbasic, an apostrophe can be used instead of **REM**.

This subroutine asks for two numbers as input and then adds them together.

```
100 REM *** INPUT AND ADD
    PRINT "WHAT ARE YOUR TWO NUMBERS"
    INPUT N1, N2
    PRINT N1, " + ";N2;" = ";N1 + N2
```

Run the program.

This subroutine rounds a number down to the nearest whole number. The *T* in line 210 stands for *total*.

Add it to the program you just ran.

```
200 REM *** ROUNDING DOWN
    T = INT(T)
```

These routines could work alone as programs, but they could also be incorporated as subroutines into a larger program with **GOSUB** and **RETURN** statements. **GOSUB** tells the computer to jump to a new line and follow instructions from there. **RETURN** tells the computer to return to the line in the main program immediately following the **GOSUB** line.

Programming Tip: Number subroutines as 100, 200 or 1000, 2000, or give them names, so they are easily identifiable in your program listing. Be sure to use an **END** statement at the end of the main program so that the computer won't drop down into its subroutines.

Now add these lines to the two subroutines. Run the program.

```
REM *** ADD AND ROUND PROGRAM ***
      CLS: PRINT "I WILL ADD NUMBERS FOR YOU"
      PRINT "AND ROUND DOWN THEIR TOTAL."
40    PRINT:GOSUB 100
      T = N1 + N2
      GOSUB 200
      PRINT "YOUR TOTAL ROUNDED DOWN IS ";T
      GOTO 40
END
←----- This is where sub 100 goes.
RETURN
←----- This is where sub 200 goes.
RETURN
```

Here is another program with a subroutine.

```
CLS
PRINT "EXAMPLE OF GOSUB...RETURN"
PRINT "USING CELSIUS AND FAHRENHEIT:"
PRINT:PRINT "IF DEGREES CELSIUS IS 15,"
C = 15
GOSUB 1000
PRINT "IF DEGREES CELSIUS IS 25,"
LET C = 25
GOSUB 1000
PRINT "IF DEGREES CELSIUS IS 30,"
C = 30
GOSUB 1000
END
1000 PRINT "DEGREES FAHRENHEIT IS";
      PRINT 9 / 5 * C + 32"."
      PRINT
      RETURN
```

Activity 2

Music is one of the things that make computers so much fun. QBasic, makes it very simple to play music on the computer.

Type in this program as an example to play a popular Christmas carol.

```
PLAY "O3 L8 EE L4 E L8 EE L4 E L8 EGDC L2 E"  
PLAY "L8 FFFFE L16 EE L8 EDDE L4 DG"  
PLAY "L8 EE L4 E L8 EE L4 E L8 EGDC L2 E"  
PLAY "L8 FFFFEE L16 EE L8 GGFD L2 C"
```

In line 10, O3 sets the octave of the music to the third octave. Octave 0 is the lowest octave, and 6 is the highest. L8 asks for eight notes (short.) This means that all the notes after that command be eighth notes until the length is changed again. The length of notes can be changed to any length from 1 (longest) to 64 (shortest). The letters, such as EE show the notes to play. In each octave, the notes are CDEFGAB, in order from lowest to highest.

There are some more commands you should be taught before you can write songs.

< or > = moves up or down one octave.

ML = plays the music legato (smooth)

MN = plays the music normally

MS = plays the music staccato (choppy music)

MF = plays the music in the foreground

MB = plays the music in the background

P1 = plays a pause with the length 1 (P2 would be pause length 2, etc.)

T1 = sets the tempo in quarter notes per minute (32 - 255)

= turns the preceding note into a sharp

- = turns the preceding note into a flat

. = plays the preceding note 3/2 as long as specified

Write a song of your own that lasts more than 30 seconds.

Activity 3

As you probably know, there are some really strange letters on the keyboard, like some of these: ` ~ | { }. Using the **CHR\$** command, you can print very letter on the keyboard plus a whole bunch more. There are 256 letters in all. Many do not show up at all.

Type in this program and run it to see them.

```
FOR A = 0 TO 255  
  C = C + 1 : IF C > 15 THEN C = 1  
  COLOR C  
  PRINT CHR$(A);  
NEXT A
```

Some of the characters (that is, letters, numbers, punctuation etc.) are box drawing characters. Try this program out to see them work.

```
PRINT CHR$(201); CHR$(205); CHR$(205); CHR$(205); CHR$(205); CHR$(187)  
PRINT CHR$(186); " XXXX"; CHR$(186)  
PRINT CHR$(186); " XXXX"; CHR$(186)  
PRINT CHR$(186); " XXXX"; CHR$(186)  
PRINT CHR$(186); " XXXX"; CHR$(186)  
PRINT CHR$(200); CHR$(205); CHR$(205); CHR$(205); CHR$(205); CHR$(188)
```

Try making a program that prints out your name in blue surrounded by a double line box.

Here is a list of all the special CHR\$ or ASCII characters and their numbers.

ASCII TABLE

00	(null)	33	!	66	B				
01	(34	A	67	C				
02)	35	#	68	D				
03		36	\$	69	E				
04	È	37	%	70	F				
05	É	38	&	71	G				
06	Í	39	>	72	H				
07	% (beep)	40	(73	I				
08	3	41)	74	J				
09	! (tab)	42	*	75	K				
10	4 (line feed)	43	+	76	L				
11	% (home)	44	>	77	M				
12	& (form feed)	45	-	78	N				
13	* (carriage return)	46	.	79	O				
14	+	47	/	80	P				
15	'	48	Ø	81	Q				
16	<	49	1	82	R				
17	=	50	2	83	S				
18	;	51	3	84	T				
19	.	52	4	85	U				
20	&	53	5	86	V				
21	Ø	54	6	87	W				
22	,	55	7	88	X				
23	0	56	8	89	Y				
24	8	57	9	90	Z				
25	9	58	:	91	[
26	6	59	;	92	\				
27	7	60	<	93]				
28	F (cursor right)	61	=	94	^				
29	: (cursor left)	62	>	95	-				
30	> (cursor up)	63	?	96	'				
31	? (cursor down)	64	@	97	a				
32	space	65	A	98	b				
99	c	132	ä	165	>Ñ	198	ƒ	231	τ
100	d	133	à	166	>0	199	ƒ	232	Φ
101	e	134	â	167	1	200	ƒ	233	Θ
102	f	135	ç	168)	201	ƒ	234	Ω
103	g	136	•	169	1	202	ƒ	235	δ
104	h	137	ë	170	5	203	ƒ	236	4
105	l	138	è	171	2	204	ƒ	237	i
106	j	139	ï	172	3	205	=	238	0
107	k	140	î	173	(206	ƒ	239	1
108	l	141	ì	174	*	207	ƒ	240	/
109	m	142	Ä	175	+	208	ƒ	241	"
110	n	143	Å	176	█	209	ƒ	242	\$
111	o	144	É	177	█	210	ƒ	243	#
112	p	145	æ	178	█	211	ƒ	244	!
113	q	146	Æ	179	┌	212	ƒ	245	"
114	r	147	ô	180	└	213	ƒ	246)

115	s	148	ö	181	⌈	214	⌈	247	.
116	t	149	ò	182	⌈	215	⌈	248	B
117	u	150	û	183	⌈	216	⌈	249	C
118	v	151	ù	184	⌈	217	⌈	250	"
119	w	152	ÿ	185	⌈	218	⌈	251	s
120	x	153	Ö	186	⌈	219	■	252	n
121	y	154	Ü	187	⌈	220	,	253	2
122	z	155	4	188	⌈	221	>	254	#
123	{	156	^	189	⌈	222	>	255	(blank >F F=)
124	:	157	-	190	⌈	223	,		
125	}	158	.	191	⌈	224	α		
126	~	159	/	192	⌈	225	β		
127	-	160	á	193	⌈	226	Γ		
128	Ç	161	í	194	⌈	227	π		
129	ü	162	ó	195	⌈	228	Σ		
130	é	163	ú	196	⌈	229	σ		
131	â	164	ñ	197	⌈	230	μ		

Activity 4

There is a whole different way to approach the problem of making pictures. Instead of using Vector Graphics, the lines circles and boxes that we have already studied, you can use Raster Graphics, or dots to make your pictures. This is done using the PSET command. PSET works just like a circle, but has

Type in the program below to try making one dot in the center of the screen white.

```
SCREEN 12  
CLS  
PSET (320,240),15
```

Using two new commands, the READ command and the DATA command, the color can be read in from a list of number stored away by the DATA command.

Try out this program.

```
SCREEN 12  
CLS  
DATA 15  
READ C  
PSET (320,240),C
```

Using looping and READ - DATA statements can produce patterns, cartoon figures and even pictures, because DATA statements can have many values.

Now type in these programs and run them.

```
SCREEN 12  
CLS  
DATA 1,2,4,1,2,4,1,2,4,1,2,4,1,2,4,1,2,4,1,2,4,1,2,4  
FOR X = 1 TO 24  
    READ C  
    PSET (X+306,240),C  
NEXT X
```

As the variable X changes in a loop, it reads a color and moves the PSET location over one. The first dot is color one and is located at (307,240). The second dot is color two and is located at (308,240), and so on.

Now run this program.

SCREEN 12

CLS

DATA 1,2,4,1,2,4,1,2,4,1,2,4,1,2,4,1,2,4,1,2,4,1,2,4

DATA 4,1,2,4,1,2,4,1,2,4,1,2,4,1,2,4,1,2,4,1,2,4,1,2

DATA 2,4,1,2,4,1,2,4,1,2,4,1,2,4,1,2,4,1,2,4,1,2,4,1

DATA 1,2,4,1,2,4,1,2,4,1,2,4,1,2,4,1,2,4,1,2,4,1,2,4

DATA 4,1,2,4,1,2,4,1,2,4,1,2,4,1,2,4,1,2,4,1,2,4,1,2

DATA 2,4,1,2,4,1,2,4,1,2,4,1,2,4,1,2,4,1,2,4,1,2,4,1

FOR Y=1 TO 6

FOR X = 1 TO 24

READ C

PSET (X+306,Y+237),C

NEXT X

NEXT Y

Index

- \$, 37
- +, 11
- , 11
- *, 11
- /, 11
- ^, 16
- <>, 26
- ABS, 17
- absolute value, 17
- addition, 12
- ALT, F N, 3, 4
- animation, 46, 47
- apostrophe, 48
- apostrophe , 48
- arrays, 47
- ASCII, 52
- back arrow key, 1
- BF, 41
- Block IF, 27-29
- box, 51
- Boxes, 41
- CASE, 29, 30
- characters, 3
- CHR\$, 51, 52
- CIRCLE, 44
- CLS, 3
- colon, 9, 26
- COLOR, 7, 9, 39-41
- colors, 40
- comma, 5
- counting, 21, 28
- Ctrl-Break, 4
- cumulative totals, 24
- deleting lines, 25
- DIM, 47
- division, 12
- ELSEIF, 28, 29
- END, 3, 5, 21, 48
- ENDIF, 27
- exponents, 16
- F5, 4
- Filled Boxes, 41
- Filling Boxes, 41
- FLASH, 9
- FOR statement, 31-33
- GET, 47
- GOSUB, 48
- GOTO, 4, 9, 31
- Graphics, 39
- IF ... THEN, 21, 26-29
- INPUT, 23
- INT, 16
- INVERSE, 9
- LEFT\$, 37, 38
- LEN, 37
- length of notes, 50
- LET statement, 19
- letters, 51
- LINE, 39, 41
- Line numbers, 4
- LOCATE, 8
- loop, 4, 31
- math operations, 11
- MB, 50
- MF, 50
- ML, 50
- MN, 50
- MS, 50
- Multiplication, 12
- Music, 50
- music in the background, 50
- music in the foreground, 50
- music legato, 50
- music normally, 50
- music staccato, 50
- nested loop, 33
- octave, 50
- Operations in parentheses, 12
- PAINT, 44
- pause, 50
- pixels, 39, 44
- PLAY, 50
- PRINT, 1, 2, 5
- PSET, 54
- PUT, 47
- quotation mark, 2, 11
- radius, 44
- Random Numbers, 18, 34-36
- RANDOMIZE TIMER, 18
- Raster Graphics, 54
- READ, 54
- REM, 48
- remark, 48
- reserved words, 19
- Resolution, 39
- RETURN, 48
- RND, 18, 34-36
- Robust Programs, 29
- SCREEN, 40
- SCREEN 12, 39, 40
- SELECT CASE, 29, 30
- semicolon, 5
- Shift-F5., 1
- Shift-F6, 1
- spaces, 14
- SQR, 17
- square root, 17
- STEP command, 31
- string variable, 37
- subroutine, 48
- subscript, 38
- subtraction, 12
- SYNTAX ERROR, 19
- TAB, 8
- tempo, 50
- variable, 19, 20, 25
- Vector Graphics, 54
- VGA graphics, 39
- XOR, 47