Time-Sharing Service

Learners Guide

Information Systems

Time-Sharing Service

Introduction to BASIC Programming

GENERAL ELECTRIC

INFORMATION SERVICE DEPARTMENT

Time-Sharing Service

INTRODUCTION TO BASIC PROGRAMMING

Learners Guide

August 1966
Reprinted ~~1-67, 6-67, 9-67,~~ 1-68, & 6-68

GENERAL ELECTRIC
INFORMATION SERVICE DEPARTMENT
203026

PREFACE

This text provides an introduction to programming techniques and to the BASIC language*. Its purpose is to lead a novice simply and easily to the point where he can write elementary programs in BASIC and can understand the more advanced material contained in other related publications. Because of this purpose, this text does not completely cover all of the capabilities of BASIC, and it does not provide details of terminal and system operation.

Related technical publications, which contain further details, are:

202026	BASIC Language: Reference Manual, which describes the BASIC language in more detail.
802207	BASIC Language Extensions: Reference Manual.
220125	BASIC Language Programming: Instructor's Guide, which is for use by instructors in teaching BASIC. Copies of available flip charts are included.
711224	Mark II BASIC Language: Reference Manual, provides the same information as 202026 for use in conjunction with Mark II.
229116A	Command System, which describes the Computer Time-Sharing Service provided by the Information Service Department of General Electric. Hardware configuration, terminal operation, and system commands are among subjects included.
711223	Mark II Time-Sharing Service: Reference Manual, provides the same information as 229116A for use in conjunction with Mark II.

The preceding, and additional manuals on FORTRAN, ALGOL, available library programs, etc., may be obtained from your local computer center or office of General Electric's Information Service Department, or from the address shown on the back cover.

* Developed by Dartmouth College

CONTENTS

1. BACKGROUND FOR UNDERSTANDING THE BASIC LANGUAGE

Before we talk about programming in BASIC we should understand what programming is. To do that, we must know something about how a computer works.

Let us begin by using a simple analogy to explain what a computer does. Suppose that you have just been on a shopping spree, and you would like to figure out how much you have spent. Since it was a rather extensive affair, it would be convenient to use a desk calculator for this figuring. Fortunately, you have a friend who owns one and knows how to run it. You pay a visit to your friend and you tell him what you have bought. The conversation might go something like this:

> "Multiply 6 pairs of shoes by the price of $15.00 and jot down the answer. Multiply 2 hats by the price of $10.00 and jot down the answer. Multiply 3 six-packs of coke by 50¢ and jot down the answer. Multiply 12 returned bottles by minus two cents and jot down the answer----and so on. Now add all the items, jot down the total and tell me what the answer is."

In this simple analogy we have outlined the elements of a simple computer such as that shown in the block diagram of Figure 1.

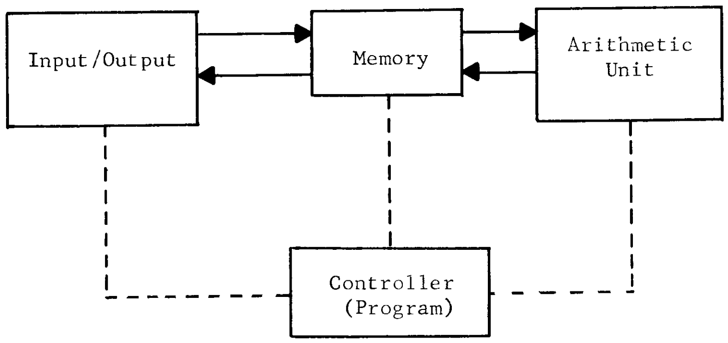

Figure 1. A Simple Computer Configuration

In our analogy, your friend's ears and his voice are the "input/output device." The scratch pad on which he jotted down your "instructions", sub-totals, and total was the memory. His desk calculator was the "arithmetic unit." His ability to understand and interpret your instructions, and to operate the desk calculator and jot down items on the scratch pad represent the "program" built into the "controller"----which is, of course, your friend.

In an actual computer these functions are all performed by electronic components. In the Computer Time-Sharing System, the processor is a GE computer, and the input/output device for our purpose is a teletypewriter. In fact, there are many teletypewriters, all connected to the same computer. The computer operates so much more quickly than you can type in or print out information that it can "time share" many input/output devices. This is done in much the same way that a good waitress in a restaurant can wait on several tables without causing any inconvenience to her other patrons. While one is drinking his cocktail, she can be serving the main course to another. While he is eating the main course, she can serve dessert to a third patron. Then she can deliver the check to a fourth in time to return to the first just as he is finishing his cocktail and is ready to order another. If the patrons were not able to see one another, each might think that the waitress was serving only him.

In the same way, the GE Time-Sharing System can accept an "order" from one teletypewriter, then service many others, and still return to the first one in time to accept the next "order" without delay. Thus, it appears to each user, sitting at his teletypewriter, that he has exclusive use of the computer. (Sometimes the computer will type out "WAIT", but will type "READY" just a few seconds later.)

The fact that a teletypewriter is used as an input/output device means that it can be located some distance from the computer. To go back to our analogy, it is as if you decided to call your friend on the telephone rather than drive over to his house. This is, of course, much more convenient. Since computers have not yet been designed to respond to the spoken word, we don't call up the computer by telephone, but by teletypewriter. The computer can readily interpret the electrical signals put out by the teletypewriter, and the human operator has little difficulty operating the teletypewriter keyboard. It is almost identical to that of a standard typewriter. By the same token, the computer can readily generate electrical signals which cause the teletypewriter to print out a message which the operator can read.

Thus, we now have all the elements of a "time-shared" computer system serving many remote locations. There remains one problem however--computers have a "language" all their own, so far as program instructions are concerned. They don't do arithmetic the way humans do, either. It is as if, in our analogy, the friend were Japanese, spoke no English, and used an abacus rather than a desk calculator. What we need is a translator. Maybe your friend's wife speaks English. We talk to her on the phone, so that she can translate into Japanese for her husband and translate his answers back into English for us. Meanwhile, he can use the abacus to do what we would have done on a calculator.

This is what a BASIC program does for the computer. The BASIC program provides translation of simple English language and mathematical instructions into computer language.

If time-sharing brings every man his own computer, the BASIC makes every man his own programmer. We do not need to concern ourselves with the fact that the computer uses binary arithmetic, rather than decimal, or that the program--as the computer sees it--is complicated by being written in binary "words".

The BASIC program will recognize many different instructions, both English language and mathematical. In time you may learn to use them all. Let's begin by seeing how BASIC, the language used by the time-sharing system, would have solved the problem of our analogy:

Our Human Method	The Computer Program
Multiply 6 by $15.00 and jot down answer.	`100 LET X1 = 6*15`
Multiply 2 by $10.00 and jot down answer.	`110 LET X2 = 2*10`
Multiply 3 by $00.50 and jot down answer.	`120 LET X3 = 3*0.5`
Multiply 12 by minus $.02 and jot down answer.	`130 LET X4 = 12*(-.02)`
Add all items and jot down answer.	`140 LET T = X1+X2+X3+X4`
Tell me what the answer is.	`150 PRINT T`
That's all, thank you.	`999 END`

While the above example is quite workable in BASIC time sharing, we have left out a few essentials, such as calling up your friend, identifying ourselves, telling him what language we would like to speak (his wife is by now multilingual), and telling him goodbye. Let's use the comparison method again to see how this works.

	HUMAN		COMPUTER
1.	Pick up phone.	1.	Push the ORIG button on teletypewriter.
2.	Listen for dial tone.	2.	Listen for dial tone on speaker.
3.	Dial friend's number.	3.	Dial the Time-Sharing computer's number.
4.	Friend answers.	4.	Computer answers (beep).
5.	Friend says, "Who is this?"	5.	Computer types USER NUMBER.
6.	Tell him your name.	6.	Type in your user number.
7.	Friend says, "What language?"	7.	Computer types SYSTEM.
8.	Tell him: "English."	8.	Type in BASIC.
9.	Friend says, "Is this a new problem or the same as before?"	9.	Computer types NEW OR OLD
10.	Tell him, "A new problem."	10.	Type in NEW.
11.	Friend says, "What kind?"	11.	Computer types NEW FILE NAME.
12.	Tell him, "Sales spree."	12.	Type in SPREE.
13.	Friend says, "O.K. let's go."	13.	Computer types READY.

Now we are ready to enter the program as previously described. Once we have entered the program, including the END statement, we simply type in RUN, and depress the RETURN key. The computer will perform the requested operations and type the answer. We perform the goodbye sequence as follows:

Say "Thank you, goodbye".	Type in BYE.
Friend hangs up.	Computer advances paper to where it can be torn off, and automatically disconnects the teleypewriter.

The following chart lists the instructions for teletypewriter operation and illustrates the keyboard.

Instructions for Teletypewriter Operation

1. Push ORIG button. This corresponds to lifting telephone off hook.

2. Listen for dial tone (SPKR VOL controls the sound you hear from speaker).

3. Dial the number of the Time-Sharing computer. If busy, push CLR button. Wait a while and try again.

4. When a computer answers with a beep, the teletypewriter will automatically type out its own identification number.

5. Type user number and other information requested, then enter a new program or call up an old one and use it.

6. When through, type BYE, followed by a carriage return. The teletypewriter will shut off automatically.

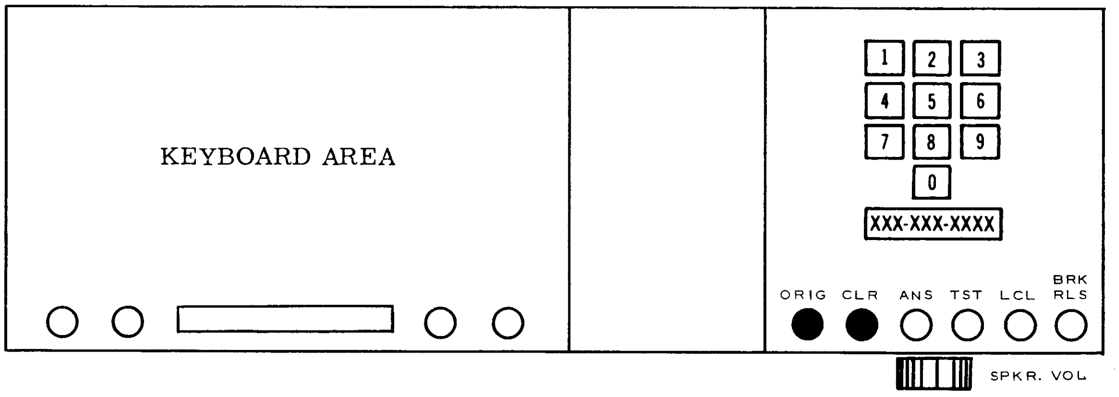

Figure 2. Teletype Unit Control Panel

Note: Controls shown solid (●) are the only controls needed for Time-Sharing operations

Here is how the entire "conversation" would appear printed on the teletypewriter:

```
USER NUMBER--W95500
SYSTEM--BASIC
NEW ØR ØLD--NEW
NEW FILE NAME--SPREE
READY.

100 LET X1=6*15
110 LET X2=2*10
120 LET X3=3*0.5
130 LET X4=12*(-0.02)
140 LET T=X1+X2+X3+X4
150 PRINT T
999 END

RUN



SPREE     21:31    W1 TUE 06/18/68

 111.26




RAN      0.33 SEC.
READY.

BYE

*** ØFF AT 21:31   ELAPSED TERMINAL TIME =      3 MIN.
```

For the purpose of illustration only, those parts of the "conversation" typed by the user have been underlined. In actual practice this underlining does not exist.

2. PREPARING A BASIC PROGRAM

Now I think we've carried our analogy far enough. It's time we entered the real world of BASIC and time-sharing. Let's discuss the BASIC program.

Each program instruction must be entered on a separate line and must be followed by a carriage return. Notice that each instruction to the machine is assigned a number---in the example we used 100, 110, 120 and so on. The computer executes the instructions in numerical order, thus the program must be written so that the instructions, when taken in numerical order, follow a logical sequence. For example, it would do no good to ask for PRINT T before the other instructions had been executed, because the computer wouldn't know the value of T and could not print it. Notice, that we said "taken in numerical order". The instructions can be typed in any order because the computer will automatically resequence them into numerical order. This is a great convenience and is one of the reasons for numbering the instructions in the first place.

If we realized after the program was all written that we had forgotten one line, we would just type in: 141 LET X5 = 3*17. All instructions would automatically be put into sequential order by the computer. We don't have to retype the whole program. For this reason, most programmers leave a few "spare" numbers between instructions. This makes it easier to correct or modify the program later. A word of caution – no more than 256 lines are permitted for any BASIC program.

Every program must have an END statement so that the computer will know when it has performed all of the program. The END statement must be the highest numbered instruction in the program. Most programmers use a sufficiently large number (for example: 999 in the example), to ensure that all other instructions will have lower line numbers. Because line numbers are limited to five digits, the largest permissible line number is 99999.

Notice that each instruction is in simple algebraic form, rather than in English or arithmetic. That is we don't say: Multiply 6 x 15. Instead we say LET X = 6*15. Any arbitrary variable may be used instead of X as long as it is a letter or a letter followed by a single digit number.

Be sure to distinguish between "zero" and the letter "oh", and between the digit "one" and the lower case "ell". There are no lower case letters on a teletypewriter, but if you are accustomed to using lower case "ell" on a typewriter for the digit "one"--watch out ! Also, because there is no lower case X, nor a multiplication sign on a teletypewriter, we use the asterisk for this purpose.

The computer is most advantageously used to do repetitive work. Repetitive, in this case, may mean doing a simple thing over and over for many different numbers---for example, find the fourth power of all the numbers from 1 to 50 in steps of 0.1. Or it may mean doing a fairly complicated operation somewhat less often, like finding the roots of a quadratic equation, with different input data each time.

3. PROVIDING DATA

Most programs can be reduced to three simple steps:

1. Providing data
2. Performing desired calculations
3. Printing out answers

Because Steps 1 and 3 are the easiest to understand, let's start with them. Assume some kind of calculation is needed, for example:

```
LET X = -3.14159 * Y
```

There are three ways in which step 1 can be performed. First, we can simply say: READ Y. Then we can include a statement: DATA 3, 7, 23, 71.

Whenever the computer encounters a READ statement in the program, it will read the next value of Y. The first time it will read 3, the next time 7, the next time 23, and so on until no more data is left to be read. Then it will stop. If our program had said: LET X = A*B, we would have had to say: READ A, B. The computer would then take the numbers two at a time from our DATA statement regarding the first number as A and the second as B, the third as A and the fourth as B and so on.

The second way of providing data is the FØR...NEXT method. For example, we could say: FØR Y = 1 TØ 10*. At the appropriate point, where the next variable is required in the program we would give a NEXT Y instruction. The computer would then perform the desired calculation for all integral values of Y from 1 to 10. If we want larger or smaller steps, we simply specify the size of step required. For example: FØR Y = 1 TØ 10 STEP 0.1 would cause the machine to use: 1, 1.1, 1.2, 1.3,... This method of input is obviously very useful for computing tables of squares, cubes, square roots, cube roots, etc.

The third way of providing data is to have the computer ask for it. For example, assume we have a program which needs the values of A, B, and C which we would like to provide each time we run the program. To do this, we use two instructions: PRINT "ENTER A, B,C" followed on the next line by INPUT A,B,C. The computer will now ask for the values of A,B and C by typing: ENTER A,B,C? Then you enter any values, for example: 2,8, -10 by typing them in and depressing the RETURN key. The computer will perform the calculations with the values you provided and print out the answer. If you have programmed a loop (which we'll get into later) it will ask you to ENTER A,B,C again.

*Notice the use of Ø for the letter "Oh" to distinguish it from the number "zero."

4. PRINTING ANSWERS

So much for providing data. Now let's see how we get information out of the computer. Again, there are three ways in which we can get the computer to print out information, using program instructions. The first two use the PRINT command. Anything included in quotes following a PRINT command will be printed as is. For example, the command PRINT "T", would cause T to be printed, perhaps as a column heading. The command PRINT T, however, without the quotes around T, would cause the computer to print the value of T. In our very first example (SPREE), for instance, we could have written 149 PRINT "T=", before the 150 PRINT T instruction. The teletypewriter would then have printed out T= on one line, and the value of T (111.26) immediately below it on the next line. To save paper, we could have written: 150 PRINT "T="; T in which case we would receive the printout: T=111.26.

Thus by using quotation marks, we can get the computer to print out what we specify, for column headings, line titles, and so on. By omitting the quotation marks we can get it to print the actual value of the quantity called for.

Now let's discuss where the computer is going to print. Each PRINT instruction causes the computer to begin a new line. A PRINT instruction with nothing after it won't print anything, but it will advance the paper one line. This is a handy technique for improving format.

The page is divided into five zones of fifteen spaces each across each line. Anything in quotes is printed just as it appears. For anything not in quotes, however, a comma is the signal to move to the next zone. For example, if we used the instruction PRINT A,B,C, (note that we used no quotation marks around the letters) the computer would start at the left edge of the page and print the value of A. Then it would skip over to column 15 and print the value of B. It would skip the necessary spaces to bring it to column 30, and print the value of C. We could thus get as many as five printouts across the page, beginning at columns 0, 15, 30, 45, and 60. This function is just like tabulating on a typewriter.

If we want to get more than five columns, we can "pack" the printouts by using a semicolon between the letters instead of a comma. Thus: PRINT A;B;C would cause the computer to print the value of A, then move to the next multiple of three columns and print the value of B, and so on. In this way, we can get many more than five printouts across the page. This is appropriate if the printouts contain only a few characters. The space between the first character of one printout and the first character of the next printout is a multiple of three, but never less than six characters. One of the difficulties in packing printouts is that the columns may not line up vertically, because the spacing between printouts depends on how many characters are printed.

How are numbers printed out? Here are the rules which the BASIC system follows:

1. No more than six significant digits are printed (except for integers---see rule 4).

2. Any trailing zeros after the decimal point are not printed.

3. For numbers less than 0.1, the form X.XXXXX E-Y is used, unless the entire significant part of the number can be printed as a six digit number. For example, 0.0000376 is the same as 3.76×10^{-5}, and will be printed as 3.76 E-5.

4. If the number is an integer (in other words a whole number), the decimal point will not be printed and up to nine digits will be printed in full.

The third method of printing out information involves the use of the LIST command which instructs the computer to type out your entire program just as it is stored in the machine. This is very useful for obtaining an up to date version of a program which has undergone many corrections, additions and deletions as it was being typed in. If LIST -- XXX is used the listing will begin at the line numbered XXX instead of at the beginning of the program.

There is another command that is useful when listing programs. This is the REM (Remarks) command. Following a REM statement the computer will type out exactly what is stated. For example:

```
190 REM THIS IS A LIST ØF CØNSTANTS.
```

The difference between PRINT and REM is that, in the first case, the words to be printed out should be in quotation marks; and they are printed when the program is RUN. In the second case, the words to be printed out are not in quotation marks; and they are printed when the program is listed, but not when it is RUN.

5. PERFORMING CALCULATIONS

Now that we know various ways of providing data, and printing answers, headings and remarks, we should turn our attention to the problem of how we get the input data computed into answers.

We have already said that program instructions should be in simple algebraic form, such as: LET X = 6*15. We can actually say: LET X = almost anything in mathematical terms. The symbols used for mathematical operations may be a little unfamiliar to you, but they result from limitations of the teletypewriter keyboard. These basic symbols are as follows:

+	plus
-	minus
*	times
/	divided by, (for example A/B means A divided by B)
↑	raise to the power of, (for example 2 ↑ 3 = 2^3 = 2x2x2)

Because each expression must be written on a single line, parentheses are often required where they might not be needed in ordinary mathematical notation. For example:

$\frac{A-B}{C}$ must be written: (A-B) / C

Omitting the parentheses would cause the expression to be interpreted as:

A - (B/C), which is not what was intended.

In addition to arithmetic operation, the following standard mathematical functions are available:

SIN (X)	Sine of X	X must be in radians
COS (X)	Cosine of X	
TAN (X)	Tangent of X	
ATN (X)	Arctangent of X	
EXP (X)	Natural exponential of X, e^X	
ABS (X)	Absolute value of X, $\|X\|$	
LOG (X)	Natural logarithm of (X)	
SQR (X)	Square root of (X)	
RND (X)	Generates a random number between zero and one at each call	
INT (X)	Integer part of X	

In any of the above expressions, X may be any quantity or expression.

6. WRITING A PROGRAM

One of the things which sometimes confuses people who are familiar with algebra but unfamiliar with programming is the use of an expression such as: LET X = X+7. Mathematically, such an expression reduces to: X-X = 7 or 0 = 7 which is meaningless. In the programming sense, however, the meaning is: Let the new value of X equal the old value of X, plus 7.

Going back to our original example of SPREE, for instance, instead of:

```
140  LET T = X1 + X2 + X3 + X4
```

we could have written:

```
140  LET T = X1
141  LET T = T + X2
142  LET T = T + X3
143  LET T = T + X4
150  PRINT T
```

This wouldn't be a very efficient way to write the program, but it does serve to illustrate the point. To show a useful purpose for such instructions, let's take a look at a program "loop".

Where the same operation is to be performed many times in the course of a program, "looping" greatly reduces the length of the program. For example, in a program to find the squares of numbers from 1 to 10 we might write:

```
100  PRINT "X" , "X ↑ 2"
110  LET X = 1
120  LET X1 = X ↑ 2
130  PRINT X , X1
140  LET X = X+1
150  GØ TØ 120
999  END
```

In this case, the program sets the initial value of X to 1, squares X, and prints X and X squared. It then adds 1 to X (making X = 2) and goes back to program step 120 to repeat the process. X is increased by 1 each time the program goes through the loop. This program would continue indefinitely, of course. We can stop it in one of two ways--either by interrupting it at the appropriate time, or by inserting another instruction as follows:

```
141  IF X = 11 THEN 999
```

or

```
141  IF X >10  THEN 999
```

In this case, when the loop has been traversed enough times so that X = 11, the program will branch to 999 (jumping over instruction 150) and stop.

Anytime we use a program loop, we must plan on a way of getting out of the loop. Otherwise, the computer will continue to run around the loop indefinitely. When printing is not taking place, we can interrupt the program by typing STØP. When printing is occurring simultaneously, depress the Control, Shift, and P keys to interrupt the program. But no program should rely on human intervention to get out of a loop. There are two customary methods for this purpose:

1. Include a READ instruction as part of the loop. The computer will use successive values from the DATA instruction lines until all values are used up, at which time it will stop, and print: ØUT ØF DATA IN XYZ (where XYZ is the line number of the READ instruction).

2. Include in the loop a "conditional branch" instruction (IF .... THEN PQR). In this case, when the given condition is satisfied, the computer will branch to the instruction whose line number is PQR. PQR may be the END statement, a STØP statement, or simply another instruction in the program, <u>outside</u> of the loop.

The condition for branching can be in the form of a test. For example: IF X = 0, IF X < 0 (if X is negative), IF X > 10, or some similar condition. We can also test to see whether X is positive by writing: IF X > 0.

The GØ TØ and IF .. THEN instructions may be used for other purposes besides forming and getting out of loops. The instruction line number specified in the GØ TØ and IF...THEN statements may cause the program to jump to either an earlier or a later instruction. For example, if the program calls for taking the square root of a negative number, the computer will perform the computation as if it were a positive number, but will automatically print out a "warning." For example: SQUARE RØØT ØF NEGATIVE NUMBER IN 50. If we don't want that warning printed out, we can avoid it in this manner:

```
040     IF P < 0 THEN 052
050     LET Q = SQR (P)
051     GØ TØ 060
052     LET Q = SQR (-P)
053     GØ TØ 060
```

In this example, if P is a positive number or zero the computer ignores 040, computes SQR (P) in 050, and goes to 060 as a result of instruction 051. If P is less than zero the program "jumps over" instructions 050 and 051 and computes SQR (-P) in step 052. Since P is negative, -P is positive, and we won't get the warning printed out. Step 053 causes us to go to 060 as before, so we end up at 060 by either route. (Actually, unless there are other intervening instructions step 053 could be omitted, since the computer would automatically arrive at 060 after 052).

7. CORRECTING A PROGRAM

To err is human----and we all make an occasional error. Here is how typing and program errors may be corrected.

To correct a line, retype it correctly using the original line number and end it with a carriage return.

To delete a line, retype the line number only, and follow it with a carriage return.

If you type an incorrect character and notice it right away, type the "backwards arrow" (←) which is found above the letter O. Then type in the correct character. This will not, of course, erase the character from the page, but it will delete the character from computer memory. For example: 120 LEX ← T X=3 will come out as "LET X=3". If you have mistyped several incorrect characters, type ← once for each incorrect character (including spaces) then type in the correct characters.

If you find you have forgotten an instruction several lines back, just type it in with the correct line number. The computer will automatically rearrange the instructions into the proper numerical sequence.

If you are a poor typist, the program may look pretty messy by the time you get through with retyped and deleted lines and backwards arrows. Don't worry about it. Just type LIST, followed by a carriage return, and the computer will print a nice clean copy of your program.

Once you are sure your program is correct, type RUN followed by a carriage return. The computer will run your program. It may type out one or more error messages for example:

NØ END INSTRUCTIØN (This is the most common one for beginners)

NØ DATA (You have put in a READ statement but forgotten to type a DATA statement)

ILLEGAL FØRMULA IN 190 (Maybe you put LET X = 5R, instead of LET X = 5*R)

Careful examination of the error messages and checking your instructions will enable you to quickly "debug" your program to the point where it will run correctly. Don't spend long periods of time at the teletypewriter just thinking or making wild guesses as to what the trouble might be. SAVE your program, type in BYE and depress the RETURN key. Then go back to your desk and analyze the situation. Once you have figured out the answer to the problem, go back to the teletypewriter, recall your program, and make corrections.

Unless you SAVE your program, the computer will "forget" it as soon as you "sign off." If you want to have the computer "remember" your program, you must type SAVE, depress RETURN, and wait for the computer to answer READY before you say BYE.

To recall a saved program, ask for an ØLD program and type in the ØLD program name. When the computer has found your program it will type READY. If you subsequently wish to UNSAVE the program, type UNSAVE and depress the RETURN key. DON'T SAVE PROGRAMS UNLESS YOU INTEND TO USE THEM AGAIN BECAUSE THEY TAKE UP STORAGE SPACE.

If you forget which program you have saved, or if you are sharing a user number and want to be sure you don't use a program name already chosen by someone else, type CATALØG, followed by a carriage return. The computer will list the names of all the programs saved under your user number.

If you want to start a new program, simply type NEW, followed by a carriage return. This may be done at any time. The computer will then ask: NEW PRØBLEM NAME--type in your new program name and new program.

Another command similar to NEW is SCRATCH. It erases everything except the name of an old program, so you can start over without renaming the program.

RENAME is another command that is useful when you wish to slightly modify an existing program, but still retain the old program. First SAVE the existing program; then type RENAME, and depress the RETURN key. The computer will ask for NEW PRØBLEM NAME. You may then type in the new name and modify your program, since the original program has been saved under the old name. Your modified program may be saved under the new name.

By now you should be able to do two things: write simple programs in BASIC and understand most of the material in the BASIC Language Reference Manual. To increase understanding, try to follow the program logic of the sample problems in the next chapter.

8. SAMPLE PROBLEMS

The diameter (D) and the area (A) of a circle are related to its radius (R) as follows:

$D = 2R$

$A = pi \times R^2 = 3.14159 \times R^2$

Write a program to compute and print out the values of D and A for various values of R. The program should print out the answer in three columns using the column headings R, D, and A respectively. The values of R, D, and A are to be printed in successive lines under the appropriate column headings.

1. Use the READ and DATA instructions and let the values of R be 1,2,3,4 and 5.
2. How would you modify the program to compute the values of D and A for R = 1 to 5 in increments of 0.1? (i.e. 1.0, 1.1, 1.2....4.8, 4.9,5.0).
3. How would you further modify the program to have the computer request values of R using the INPUT command?

ANSWERS TO SAMPLE PROBLEMS

Problem #1

```
010 PRINT "THIS PRØGRAM CØMPUTES THE DIAMETER AND AREA"
020 PRINT "ØF A CIRCLE AS A FUNCTIØN ØF THE RADIUS."
030 PRINT
040 PRINT "R", "D", "A"
050 READ R
060 LET D=2*R
070 LET A=R↑2*3.14159
080 PRINT R, D, A
090 GØ TØ 050
100 DATA 1,2,3,4,5
999 END
```

When this program is run, the following is printed:

```
CIRC-1     14:18

THIS PRØGRAM CØMPUTES THE DIAMETER AND AREA
ØF A CIRCLE AS A FUNCTIØN ØF THE RADIUS.

R                D                A
 1                2                3.14159
 2                4                12.5664
 3                6                28.2743
 4                8                50.2654
 5                10               78.5398

ØUT ØF DATA IN  50
```

Problem #2

Rename the program calling it CIRC-2, and make the following changes:

```
050 FØR R=1 TØ 5 STEP .1
090 NEXT R
100
```

Following is a listing of CIRC-2 after the changes have been made:

```
CIRC-2     14:20

010 PRINT "THIS PRØGRAM CØMPUTES THE DIAMETER AND AREA"
020 PRINT "ØF A CIRCLE AS A FUNCTIØN ØF THE RADIUS."
030 PRINT
040 PRINT "R", "D", "A"
050 FØR R=1 TØ 5 STEP .1
060 LET D=2*R
070 LET A=R↑2*3.14159
080 PRINT R, D, A
090 NEXT R
999 END
```

If the program is run, the following is printed:

```
CIRC-2     14:21

THIS PRØGRAM CØMPUTES THE DIAMETER AND AREA
ØF A CIRCLE AS A FUNCTIØN ØF THE RADIUS.

R               D               A
 1               2               3.14159
 1.1             2.2             3.80132
 1.2             2.4             4.52389
 1.3             2.6             5.30929
 1.4             2.8             6.15752
 1.5             3.              7.06858
 1.6             3.2             8.04247


STØP.
READY.
```

The program would have continued in steps of .1 through R=5, but we depressed Control, Shift, and P as soon as we saw it was working properly.

Problem #3

Rename the program calling it CIRC-3, and make the following changes:

```
025 PRINT "YØU ENTER AS REQUESTED."
040
050 PRINT "ENTER R";
055 INPUT R
080 PRINT "R=";R,"D=";D,"A=";A
090 GØ TØ 030
```

Following is a listing of CIRC-3 after the changes have been made:

```
CIRC-3     14:30

010 PRINT "THIS PRØGRAM CØMPUTES THE DIAMETER AND AREA"
020 PRINT "ØF A CIRCLE AS A FUNCTIØN ØF THE RADIUS."
025 PRINT "YØU ENTER AS REQUESTED."
030 PRINT
050 PRINT "ENTER R";
055 INPUT R
060 LET D=2*R
070 LET A=R↑2*3.14159
080 PRINT "R=";R,"D=";D,"A=";A
090 GØ TØ 030
999 END
```

If the program is run, the following is printed:

```
CIRC-3     14:31

THIS PRØGRAM CØMPUTES THE DIAMETER AND AREA
ØF A CIRCLE AS A FUNCTIØN ØF THE RADIUS.
YØU ENTER AS REQUESTED.

ENTER R? 1
R= 1            D= 2            A= 3.14159

ENTER R? 2
R= 2            D= 4            A= 12.5664

ENTER R? 3
R= 3            D= 6            A= 28.2743

ENTER R? 9.75
R= 9.75         D= 19.5         A= 298.647

ENTER R? STØP



STØP.
READY.
```

APPENDIX
SAVING PROGRAMS ON PUNCHED PAPER TAPE

You can save programs by storing them on the system as described previously. Storage space is not unlimited, and it is strongly recommended that programs which are to be used infrequently be saved by the user himself on punched paper tape. They can then be entered into the system very quickly when needed.

It is also possible to prepare programs on tape when the teletypewriter is in the LCL (local) mode of operation and not connected to the computer. But regardless of whether the paper tape was prepared while listing a program or using the teletypewriter off-line from the computer, there are some precautions to observe.

To punch a program into paper tape:

1. <u>In the Local Mode</u>. The procedure to punch a program into tape while not connected to the Time-Sharing system is simple. Press the LCL button on the teletypewriter. Turn the tape punch on. Punch 10-15 RUBOUT characters, then punch the program. Be sure to punch <u>only</u> program lines, not the program name, date, etc. At the end of each program line, enter carriage RETURN, a LINE FEED, and a RUBOUT. When you have punched the program, punch 10-15 RUBOUT characters.

2. <u>In the Time-Sharing Mode</u>. This is easier, if you want to punch into tape a saved program. Call up the program. When READY types, type in the command LISTNH but do not return the carriage. Turn the tape punch on, punch in 10-15 RUBOUT characters, and then press RETURN. The program will be listed without a heading and at the same time punched in the tape. When the program is completely listed, punch 10-15 more RUBOUT characters.

3. <u>Correction of Errors</u>. You may need to correct errors when punching tape in the local mode. To do so, use the ← key and the ALT MODE key (or its equivalent) to delete characters or lines. Or you can backspace the paper tape to the mistake and punch RUBOUT characters over the error. The computer will respond to these corrections when the tape is read in.

To load into the computer a program stored on paper tape:

1. Place the tape into the tape reader by pushing the "release" button. Raise the tape guide and lay the tape carefully (right side up) into the tape slot. Be sure that the <u>feed</u> pins fit properly in the <u>feed</u> holes. Then, holding the tape carefully in place, lower the tape guide on to it. Locate the tape so that the tape read pins are in a position to read a few RUBOUT characters ahead of the program itself.

2. Go through the usual preliminary sequence. Enter NEW problem and the name of your program as requested. When the teletypewriter types READY, type in TAPE followed by a carriage return.

3. When the teletypewriter types READY again, turn the tape reader ON. Be sure you have prepared a "leader" for your tape as described above.

4. The tape reader will now read your tape into the computer, simultaneously printing it out on the teletypewriter (and also punching a duplicate tape if your tape punch is ON). Be sure to keep the tape from twisting or looping as it feeds through the tape reader. When the end of the tape is reached, the tape reader will stop automatically.

5. Your program is now entered into the computer. Type KEY (for keyboard) followed by a carriage return. When the computer answers READY, proceed just as if you had entered the program manually or had recalled a saved program.

Computer Centers and offices of the Information Service Department are located in principal cities throughout the United States.

Check your local telephone directory for the address and telephone number of the office nearest you. Or write . . .

General Electric Company
Information Service Department
7735 Old Georgetown Road
Bethesda, Maryland 20014

GENERAL ELECTRIC

INFORMATION SERVICE DEPARTMENT

203026 (20M) 6-68