SCELBI'S

SECRET GUIDE TO COMPUTERS

QUICKIE COURSE ON COMPUTERS

- Become a computer expert
- Learn how to do it in 20 minutes
- Discover BASIC the easy way
- Log your own "secret" programs

Free Bonus:
Your personal questions answered by telephone, day or night, 24 hours.
Secret telephone number provided.

SCELBI Publications

SCELBI's Secret Guide to Computers

By Russell Walter

SCELBI Publications

Foreword

Twenty minutes from now, you'll know how to run a computer. If you have a computer in front of you, it will already be obeying your commands. You don't have to be brilliant. I've taught seven-year-old kids how to do it.

You don't have to know any math. If you know what decimals are— if you know that 6.2 is between 6 and 7— you know more than enough math to get through this book, because most computer applications have little to do with math. "Programming a problem for a computer" does *not* mean "reducing everything to numbers"; that idea is totally false. Programming a problem for a computer means: reducing a big problem to a chain of little ones. The little ones don't have to involve numbers. The computer can handle words as well as numbers. The term "computer" is an anachronism: the French call the machine "l'ordinateur", "the organizer", which describes it better.

Yes, mathematicians program computers more easily than non-mathematicians. So do lawyers. Anybody who's been trained to reduce a big problem to a chain of little ones has an advantage. If you haven't had that kind of training yet, this book is your big chance. You'll learn how to "think logically". Once you learn that skill, it will help you in *all* aspects of your life— not just in programming.

Russell Walter

Secret Telephone Number

617-266-8128

Whenever you have a question about computers, pick up your phone and give me a buzz. No matter how weird or personal your question, you'll get an answer, free! Call as often as you like; ask *all* your questions. You can call day or night—24 hours! I'm usually in, I don't sleep much, and I sleep only lightly. This free consulting service has saved readers many kilohours and kilobucks.

I'm at (617) 266-8128, Boston. If you live elsewhere, remember that the phone company has lowered the rates: for example, to call all the way from California to Boston costs *just 5¢ initially, plus 16¢ per minute.* To get that low rate, call without using an operator, and call anytime Saturday, or on Sunday before 5PM, or on weekdays before 8AM or after 11PM. If one of my cohorts answers and I'm not in, leave your number, and I'll call you back.

I'm waiting to hear your questions!

Russell Walter
92 Saint Botolph St.
Boston, MA 02116

Contents

Model Programs

The only way to learn programming is to look at sample programs, analyze them, and then invent your own. This volume contains 150 sample programs, analyzes them, and gives you hints on how to invent your own.

Of the programs in this book, 24 are particularly important: they are the "models"; the other programs just elaborate on the models. Each lesson contains 6 models.

Here are the models. You don't have to look at them now; but later, when you're in the middle of writing your own programs, you'll want to refer back to this page, for help.

LESSON 1: CHAT WITH YOUR COMPUTER

Printing - page 1-6

```
1 PRINT "I LOVE YOU"
2 PRINT "YOU TURN ME ON"
```

The computer will print:

```
I LOVE YOU
YOU TURN ME ON
```

Semicolon - page 1-10

```
1 PRINT "FAT";
2 PRINT "HER"
```

The computer will print FAT and HER on the same line:

```
FATHER
```

Embedded semicolon - page 1-10

```
1 PRINT "SIN";"KING"
```

The computer will print SIN and KING on the same line:

```
SINKING
```

Go to - page 1-12

```
10 PRINT "CAT"
20 PRINT "DOG"
30 GO TO 10
```

The computer will print CAT and DOG repeatedly.

Arithmetic - page 1-16

```
10 PRINT 2+2
```

The computer will compute 2+2 and print the answer:

```
 4
```

Order of operations - page 1-17

```
10 PRINT 2+3*4
```

The computer will start with 2, then add three fours. It will print:

```
 14
```

LESSON 2: MAKE YOUR COMPUTER THINK

Variable - page 2-1

```
10 X=47
20 PRINT X+2
```

Since X is 47, line 20 says to print 47+2. The computer will print:

```
 49
```

String variable - page 2-3

```
10 G$="DOWN"
20 PRINT G$
```

Since G$ is "DOWN", line 20 prints:

```
DOWN
```

Input - page 2-4

```
10 INPUT "WHAT IS YOUR NAME";N$
20 PRINT "I HATE THE NAME ";N$
```

Line 10 makes the computer ask—

```
WHAT IS YOUR NAME?
```

and then wait for you to answer; your answer will be called N$. If you answer JOEY, line 20 will print:

```
I HATE THE NAME JOEY
```

Stop - page 2-13

```
10 PRINT "BUBBLE"
20 STOP
30 PRINT "FOX"
```

The computer will print just:

```
BUBBLE
```

Colon - page 2-13

```
10 A=5: B=7: PRINT A+B
```

The computer will print:

```
 12
```

If - page 2-13

```
10 INPUT "ARE YOU MALE OR FEMALE";A$
20 IF A$="MALE" THEN PRINT
   "SO IS FRANKENSTEIN": STOP
30 IF A$="FEMALE" THEN PRINT
   "SO IS MARY POPPINS": STOP
40 PRINT "PLEASE SAY 'MALE'
   OR 'FEMALE'": GO TO 10
```

Line 10 asks whether you're male or female. After you answer, the

computer will print an appropriate retort.

LESSON 3: MASTER YOUR COMPUTER

Data - page 3-5

```
10 DATA MEAT,POTATOES,
   LETTUCE
20 READ A$
30 PRINT A$
40 GO TO 20
```

The computer will print:

```
MEAT
POTATOES
LETTUCE
OUT OF DATA
```

Restore - page 3-6

```
10 DATA MEAT,POTATOES,LETTUCE,END
20 READ A$: IF A$="END" THEN PRINT
   "THOSE ARE MY FAVORITE FOODS":
   RESTORE: GO TO 20
30 PRINT A$
40 GO TO 20
```

The computer will print MEAT, POTATOES, LETTUCE, and THOSE ARE MY FAVORITE FOODS, repeatedly.

For - page 3-11

```
10 FOR I = 1 TO 100
20     PRINT I
30 NEXT I
```

The computer will print every number from 1 to 100.

Step - page 3-17

```
10 FOR I = 10 TO 1 STEP -1
20     PRINT I
30 NEXT I
40 PRINT "BLAST OFF!"
```

The computer will print a rocket countdown: 10, 9, 8, 7, 6, 5, 4, 3, 2, 1, and BLAST OFF!

Random - page 3-17

```
10 RANDOM
20 PRINT RND(5)
```

The computer will print 1 or 2 or 3 or 4 or 5.

Comma - page 3-22

```
10 PRINT "SIN","KING"
```

The computer will print SIN in the first zone, and KING in the second:

```
SIN             KING
```

LESSON 4: TACKLE THE TOUGH STUFF

Recursion - page 4-1

```
10 A=5
20 A=3+A
30 PRINT A
```

Line 20 means: the new A is 3 plus the old A. So the new A is 3+5. Line 30 prints:

```
 8
```

Counting - page 4-2

```
10 C=3
20 PRINT C
30 C=C+1
40 GO TO 20
```

The computer will count, starting at at 3. It will print 3, 4, 5, 6, etc.

Summing - page 4-6

```
10 S=0
20 PRINT "NOW THE SUM IS";S
30 INPUT "WHAT NUMBER DO YOU
   WANT TO ADD TO THE SUM";X
40 S=S+X
50 GO TO 20
```

The computer will imitate an adding machine.

Subscripts - page 4-10

```
10 DIM X(3)
20 X(1)=57.2
30 X(2)=-8.3
40 X(3)=476
50 MAT PRINT X
```

Line 10 says X will be a list of 3 numbers. Lines 20-40 define them. Line 50 prints them:

```
 57.2
-8.3
 476
```

Mat read - page 4-11

```
10 DIM X(3)
20 DATA 57.2, -8.3, 476
30 MAT READ X
40 MAT PRINT X
```

Line 10 says X will be a list of 3 numbers. Line 30 reads them from the data. Line 40 prints them:

```
 57.2
-8.3
 476
```

Double subscripts - page 4-14

```
10 DIM X(2,3)
20 X(1,1)=57
30 X(1,2)=8.4
40 X(1,3)=-6
50 X(2,1)=1000
60 X(2,2)=0
70 X(2,3)=7.77
80 MAT PRINT X
```

Line 10 says X will be a table having 2 rows and 3 columns. Lines 20-70 tell which numbers are in the table. Line 80 prints the table.

Lesson 1

Chat with Your Computer

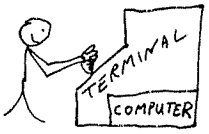
Your First Step

The Magic Typewriter

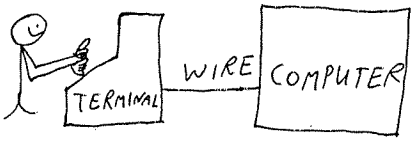
The computer is fun, because of the magic typewriter that comes with it. To boss the computer, you put your fingers on the typewriter and type your commands. Presto! the computer obeys, because the computer is attached to the typewriter.

The typewriter is called a *terminal*. If your terminal is old-fashioned, it uses paper, like an ordinary typewriter. But if your terminal is fancy-shmancy new, it uses a television screen instead; whatever you type appears on television. Groovy!

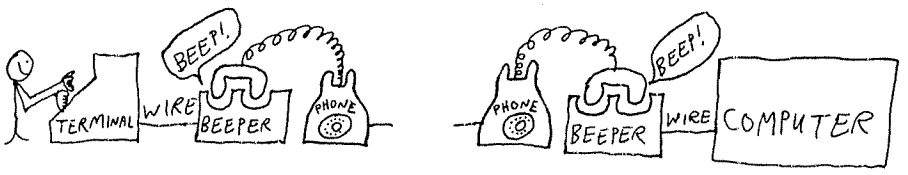
To attach the computer to the terminal, you can use three methods. If your computer is a very small microcomputer, hide the computer inside the terminal:



If your computer is too big to fit inside the terminal, put the computer *next* to the terminal, and run a wire from the terminal to the computer:



If you want to use a computer that's far away, attach your terminal to a beeper (called an *acoustic coupler* or *modem*), and make sure the computer is attached to another beeper; when the terminal and the computer want to communicate, they beep to each other over the telephone:



Be Bold

In science fiction, computers blow up. In real life, they never do. No matter what buttons you press, no matter what commands you give, you won't hurt the computer. The computer is invincible! So go ahead and experiment. If it doesn't like what you say, it will gripe at you, but so what?

When you try using the computer, you'll have trouble. It might be because you're making a mistake. It might be because the computer is broken. Or it might be because your computer is weird, and works differently from the majority of computers described in this book. (Each computer has its own "personality", its own quirks.)

Whenever you have trouble, laugh about it, and say, "Oh, boy! Here we go again!" (If you're Jewish, you can say all that more briefly, in one word: "Oy!") Then get some help.

If you're in a computer center, and other people are nearby, ask them for help. They'll gladly answer your questions, because they like to show off, and because the way *they* got to know the answers was by asking. Computer folks like to explain computers, just as priests like to explain religion. Remember: you're joining a cult! Even if you don't truly believe in "the power and glory of computers", at least you'll have a few moments of weird fun. So play along with the weird computer people, boost their egos, and they'll help you get through your initiation rite. Above all, assert yourself, and *ask questions*. "Shy guys finish last."

When dealing with the computer and the people who surround it, be friendly but also aggressive. To make sure you get your money's worth from a computer course, ask your teacher and coworkers questions, questions, questions!

If you're using a computer that you yourself own, get help from the person you bought it from.

Your town probably has a "computer club". (To find out, ask the local schools and computers stores.) Join the club, and make an announcement that you'd like help with your computer. Probably some computer hobbyist will help you.

And don't forget— you can always call *me*, anytime you want free help.

How to Get Started

(in case you're one of those lucky critters who has a computer)

Before you start playing with the terminal, make sure nobody else is using it. If nobody's sitting at the terminal, probably nobody's using it. But you can't be sure. Maybe somebody *is* using it, and just went out for a few seconds, to get a drink of water or go to the bathroom. To find out whether somebody's using it, ask the folks nearby.

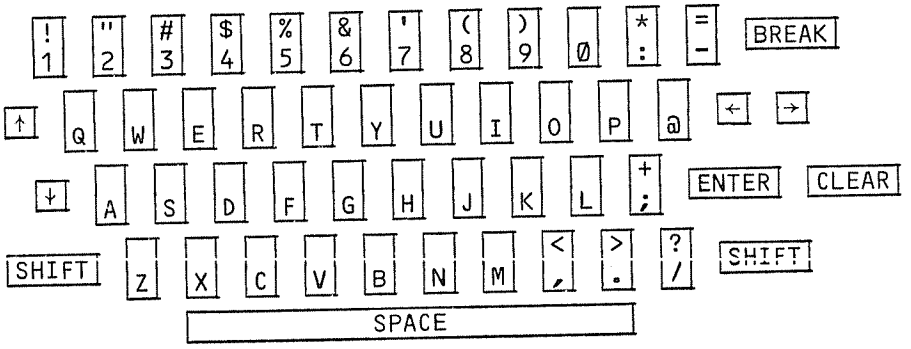
Make sure the terminal's turned on. (If it has a knob marked ON-OFF, turn it to ON. If the knob's marked LOCAL-LINE-OFF instead, turn it to LINE.)

Make sure the terminal's attached to the computer. (If you're using a beeper, turn it on, pick up the phone, dial the computer's phone number, and wait for the computer to answer the phone. It will answer by making a high-pitched hum. Then put the phone's receiver onto the beeper. Make sure the receiver's cord-end is in the correct hole of the coupler.)

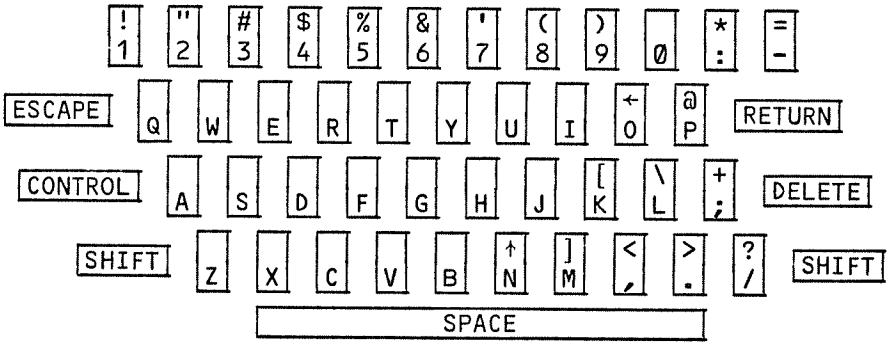
Famous Keyboards

Like a typewriter, your terminal has a keyboard. Here are pictures of the most famous keyboards.

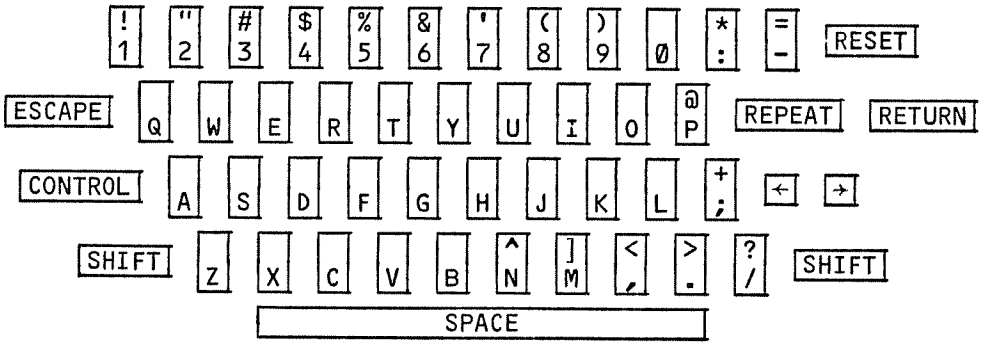
Radio Shack's keyboard



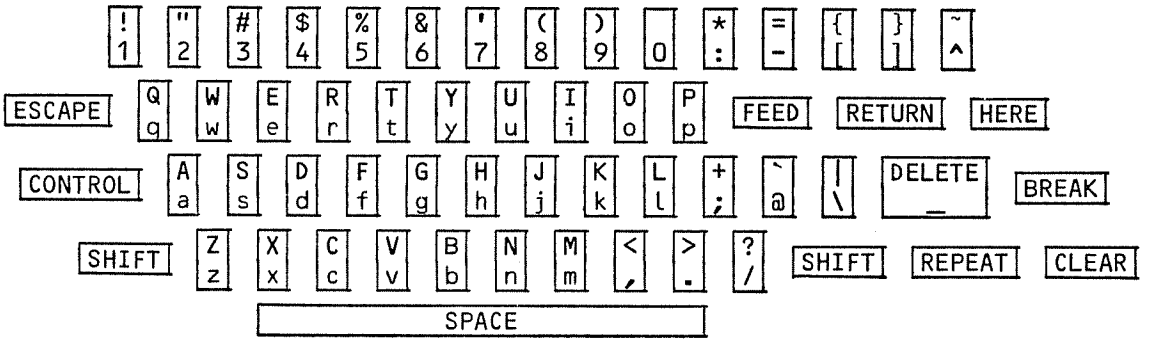
The traditional keyboard



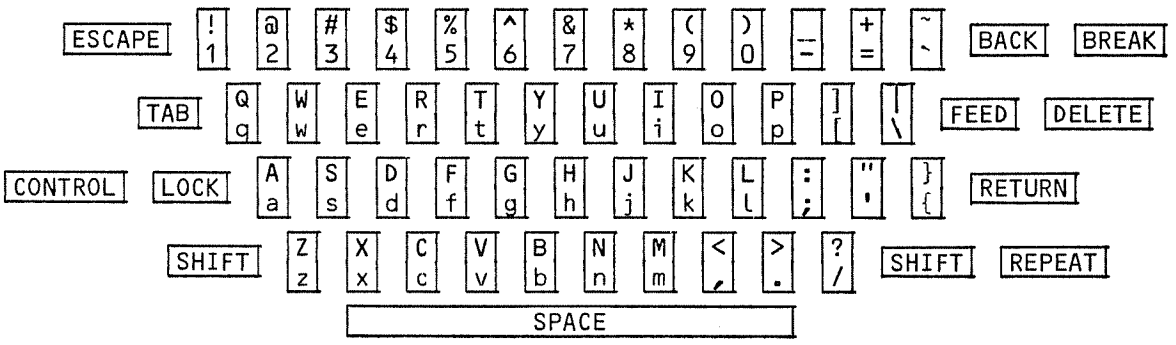
Apple's keyboard



Lear Siegler's ADM-3 keyboard



The Decwriter's keyboard



How to Use the Keys

Here's how to use the keys. (But don't touch them, until you've read the next section, which is called GETTING INTO BASIC.)

Numbers

The top row of keys contains the numbers. Don't confuse 1 with I; the number 1 is in the top row. Don't confuse zero with the letter O; the number zero is in the top row, and might have a slash through it, like this: Ø. A zero that has a slash through it is called a "Swedish zero". Look at *your* terminal, and see whether its zero is Swedish.

The Shift Key

One of the keys looks like this: $\boxed{\begin{matrix}\$\\4\end{matrix}}$. If you press it, you'll type a 4.

Suppose you hold down the SHIFT key; and while you keep holding down the SHIFT key, press the $\boxed{\begin{matrix}\$\\4\end{matrix}}$ key. When you do, you'll be typing the dollar sign.

Here's the general rule: if a key has two symbols on it, and you want to type the top symbol, hold down the SHIFT key.

About the SHIFT key, beginners often make two boo-boos. The first boo-boo is to forget to press it. For example, suppose you want to type a dollar sign, and you press the $\boxed{\begin{matrix}\$\\4\end{matrix}}$ key, but you forget to hold down the SHIFT key. Then you'll be typing a 4 instead.

The other boo-boo is to try to hit the SHIFT key and another key at exactly the same time. You can't do it; it's impossible; you'll wind up hitting one key before the other. The trick is to hold down the SHIFT key *first*; and while you keep holding it down, give the other key a light tap.

Beginners often make those boo-boos. When they realize their mistakes, they say "Oh, s...!" That's why programmers call it the "s..." key.

On Radio Shack's keyboard, and the traditional keyboard, and the Apple keyboard, the letters are automatically capitalized; so to type a capital R, just press the R key. On Lear Siegler's ADM-3 keyboard and the Decwriter's keyboard, the letters are *not* automatically capitalized; if you press the R key, you'll be typing a small r; to type a capital R, you must hold down the SHIFT key (or the LOCK key).

The Return Key

The most important key is the RETURN key. (On some keyboards, it's called the CARRIAGE RETURN or CR or RET or NEW LINE. On Radio Shack's keyboard, it's called ENTER.) Press the RETURN key at the end of every line you type. The RETURN key makes the computer read what you typed. The computer ignores what you type, until you press the RETURN key. If you forget to press the RETURN key at the end of the line, the computer doesn't read what you typed, and so the computer doesn't do anything, and you wonder why the computer seems broken.

Getting into BASIC

After you've stared at the keyboard, you must *get into BASIC*. Here's how.

If You Have a Microcomputer

Turn the computer on. Here's what happens next, on various computers.

PET and CBM computers. The computer will say "READY".

Apple 2-plus computers. The computer will print the symbol "]".

Apple 2 computers (having Applesoft in ROM). Press the RESET key. The computer will print an asterisk. Hold down the CONTROL key; and while you keep holding down the CONTROL key, tap the B key. Press the RETURN key. The computer will print the symbol "]".

Radio Shack computers. The computer will print a message. If the message says "DOS READY", type the word BASICR, then press the ENTER key. If the

computer asks "HOW MANY FILES?", press the ENTER key. If the computer asks "MEMORY SIZE?", press the ENTER key. Finally, the computer will say "READY".

If Your Have a Minicomputer or Maxicomputer

Get an *identification number*, from the people who run the computer center. When the computer asks you to type the identification number, make sure you type it correctly. If it contains a zero, make sure you type a zero, not the letter O. If it contains a comma, make sure you type the comma, and do *not* put a space before or after the comma.

To use the computer, you might also have to know a secret password; ask the people who run your computer center. If the computer asks you for the password, and you type it, the letters will be invisible; so if your enemy's looking over your shoulder, he won't be able to read it.

Here are the details, for various computers. I'll draw a black box (■) wherever you're supposed to press the RETURN key.

Honeywell computers (using DTSS). The computer will say "USER NUMBER"; type your user number (which is probably your student ID number). ■ The computer will say "PASSWORD"; type your password (which will be invisible). ■ The computer will say "NEW OR OLD".

Nova and Eclipse computers (using DOS). The computer will print the letter R; type the word DOSBASIC. ■ The computer will say "ACCOUNT NAME"; type your account's name or number. ■ The computer will print an asterisk.

Nova and Eclipse computers (using RDOS). The computer will say "BASIC" and "READY"; press the DELETE key. The computer will say "ACCOUNT NAME"; type your account's name or number. ■ The computer will say "PASSWORD"; type your password (which will be invisible). ■ The computer will print a message saying "SIGN ON"; then it will print an asterisk.

HP-2000 computers. ■ The computer will say "PLEASE LOG IN"; type the word HELLO, a hyphen, your identification number, and a comma; then hold down the CONTROL key; while you keep holding down the CONTROL key, type your password (which will be invisible). ■ The computer will print a message; then it will say "READY".

PDP-11 computers (using BASIC-11). The computer will say "PLEASE TYPE IN HELLO"; type the word HELLO. ■ The computer will say "USERID"; type your identification number or identification name. ■ The computer will say "PASSWORD"; type your password (which will be invisible). ■ The computer will say "WELCOME TO MU BASIC-11/RT-11".

PDP-11 computers (using BASIC-PLUS). Type the word HELLO. ■ The computer will print a message, then the symbol "#"; type your identification number. ■ The computer will say "PASSWORD"; type your password (which will be invisible). ■ The computer will say "READY".

PDP-20 computers. Hold down the CONTROL key; while you keep holding down the CONTROL key, tap the C key. The computer will print the symbol "@"; type the word LOG. ■ The computer will say "USER"; type your last name. ■ The computer will say "PASSWORD"; type your password (which will be invisible). ■ The computer will say "ACCOUNT #"; type your account number (identification number). ■ The computer will print the word JOB, a message, and the symbol "@"; type the word BASIC. ■ The computer will say "READY".

PDP-10 computers. Hold down the CONTROL key; while you keep holding down the CONTROL key, tap the T key. If nobody else is using that terminal, the computer will say "THIS TERMINAL IS FREE", and print a period underneath; after that period, type the word LOG. ■ The computer will print the word JOB, a message, and the symbol "#"; type your identification number. ■ If the computer says "PASSWORD", type your password (which will be invisible), and press the RETURN key. The computer will print today's message (which you can usually ignore), and then a period; after the period, type the letter R, then a space, then the word BASIC (like this: R BASIC). ■ The computer will say "READY".

CDC computers. The computer will say "USER NUMBER"; type your user number. ■

The computer will say "PASSWORD"; if you have a password, type it (invisibly); if you don't have a password, don't type anything. ■ If the computer says something about "IDNUM", type the word IDNUM, a comma, and your ID number; then press the RETURN key. If the computer says something about "CHARGE", type the word CHARGE, a comma, and your charge number; then press the RETURN key. The computer will finally say "SYSTEM"; type the word BASIC. ■ The computer will say "OLD, NEW, OR LIB FILE".

Programming

Now that you've gotten into BASIC, you're ready to program the computer! Programming the computer consists of three steps....

Step 1: type the word NEW. That tells the computer you're going to invent a new program. After you type the word NEW, press the RETURN key. The computer will say READY.

Step 2: type your program. A *program* is a list of numbered commands. For example, suppose you want the computer to say:

```
I LOVE YOU
YOU TURN ME ON
LET'S GET MARRIED
```

Type this program:

```
1 PRINT "I LOVE YOU"
2 PRINT "YOU TURN ME ON"
3 PRINT "LET'S GET MARRIED"
```

Every line of the program must be numbered; you must type those numbers. Every line of the program must say PRINT; you must type the word PRINT. Every line of the program must have quotation marks; you must type the quotation marks. At the end of each line, press the RETURN key.

Step 3: type the word RUN. That tells the computer to look at the program you've written, and run through it. After you type the word RUN, press the RETURN key. The computer will print everything you said; it will print:

```
I LOVE YOU
YOU TURN ME ON
LET'S GET MARRIED
```

Then the computer will say READY.

Afterwards, you can feed the computer another program, by going through the three steps again: type the word NEW, type the program, and type the word RUN. Remember to press the RETURN key at the end of each line.

Those three steps are all you have to know about programming! Go ahead, and write some programs! Try it; you'll have lots of fun! A person who writes a program is called a *programmer*. Congratulations: *you're* a programmer!

Weird Computers

On HP-2000 computers, say SCRATCH instead of NEW. On Compucolor computers, do this instead of typing the word NEW: while you hold down the CONTROL and SHIFT keys, tap the CPU RESET key.

When you've typed the word NEW, some computers say FILE NAME (or NEW FILE NAME) instead of READY. To respond, type the letter A, and press the RETURN key; then the computer will say READY.

On PDP-8, PDP-10, HP-2000, Honeywell, and Univac computers, the program's bottom line must say END, like this:

```
1 PRINT "I LOVE YOU"
2 PRINT "YOU TURN ME ON"
```

```
3 PRINT "LET'S GET MARRIED"
4 END
```

Multiple Copies

Suppose you want to make multiple copies of this poem:

```
SPRING HAS SPRUNG.
THE GRASS HAS RIZ.
I WONDER WHERE
THE FLOWERS IS.
```

First, turn the poem into a program, by typing:

```
NEW
1 PRINT "SPRING HAS SPRUNG."
2 PRINT "THE GRASS HAS RIZ."
3 PRINT "I WONDER WHERE"
4 PRINT "THE FLOWERS IS."
```

Then type the word RUN (and press the RETURN key). The computer will print the poem. If you type the word RUN again, the computer will print the poem again. Each time you type the word RUN, the computer will print the poem. So to print many copies of the poem, just type the word RUN many times, like this:

You type→→→→→→→→→→→→RUN

The computer types→→SPRING HAS SPRUNG.
THE GRASS HAS RIZ.
I WONDER WHERE
THE FLOWERS IS.
READY

You type→→→→→→→→→→→→RUN

The computer types→→SPRING HAS SPRUNG.
THE GRASS HAS RIZ.
I WONDER WHERE
THE FLOWERS IS.
READY

You type→→→→→→→→→→→→RUN

The computer types→→SPRING HAS SPRUNG.
etc.

If you hate poetry and all other forms of art, don't despair: by using the same technique, you can make the computer type many copies of your favorite screwed-up business letter.

Errors

How to Cancel a Character

A *character* is a letter or a digit or a punctuation mark or any other symbol you can type. If your finger slips, and you type the wrong character, here's how to cancel the last character you typed:

Computers	How to cancel the last character you typed
PET, CBM	Press the key that says DEL.
Radio Shack, Apple	Press the key that has a left-arrow (←).
PDP-8, HP-2000	Type a left-arrow (←).
IBM (using CMS)	Type the symbol @.
Honeywell (DTSS)	While you hold down the CONTROL key, tap the Z key.
CDC	Press the key that says BACK or BACKSPACE. (If your

terminal doesn't have such a key, do this: while you hold down the CONTROL key, tap the H key.)

Most others (including PDP-10, PDP-11, PDP-20, Nova, Eclipse): Press the key that says DEL or DELETE or RUBOUT. (If you're using Lear Siegler's ADM-3 terminal, the word DELETE is on the *top* part of the key, so you must simultaneously hold down the SHIFT key.)

For example, if you type the word SALE and then cancel the last character you typed, you'll cancel the E, so you'll be left with SAL.

When you cancel a character, what happens to it? If your computer is modern, the cancelled character "vanishes" from the television screen. If your computer is old-fashioned, the cancelled character remains on the screen (or paper), but is "ignored" by the computer. To emphasize which character it's ignoring, the computer might print a backslash (\) and a *second* copy of the character.

To cancel the last *two* characters you typed, do the same procedure *twice*. For example, if you type the word MOTHER and then "cancel the last character you typed" *twice*, you'll be cancelling the ER, so you'll be left with MOTH. (Then, if you type BALLS, you'll have MOTHBALLS.)

How to Correct a Mistake

If you make a mistake, you can correct it in three ways....

If you notice the mistake before you press the RETURN key, correct the mistake by cancelling the character.

If you mess up the whole program, rewrite it, by typing the word NEW and then starting from the beginning again.

If you mess up a line, retype it underneath. For example, suppose you type—

```
1 PRIMT "I LOVE YOU"
```

and then you notice that you misspelled the word PRINT. Just retype the line, so your screen or paper looks like this:

```
1 PRIMT "I LOVE YOU"
1 PRINT "I LOVE YOU"
```

Here's another example....Suppose you type—

```
1 PRINT "I LIKE COFFEE"
2 PRINT "I LIKE TEE"
3 PRINT "I LIKE THE GIRLS"
4 PRINT "AND THE GIRLS LIKE ME"
```

and then you notice you misspelled the word TEA in line 2. Underneath the program, retype that line, so your screen or paper looks like this:

```
1 PRINT "I LIKE COFFEE"
2 PRINT "I LIKE TEE"
3 PRINT "I LIKE THE GIRLS"
4 PRINT "AND THE GIRLS LIKE ME"
2 PRINT "I LIKE TEA"
```

Gripes

If the computer gripes, don't worry: just correct your mistake. For example, suppose you type—

```
1 PRIMT "ROSES ARE RED"
2 PRINT "CABBAGE IS GREEN"
3 PRINT "MY FACE IS FUNNY"
4 PRINT "BUT YOURS IS A SCREAM"
```

and you don't notice that you misspelled PRINT in line 1. If you type RUN, the computer will try to run the program, but will be confused by line 1, so it will gripe at you. It will say something like:

```
YOU MADE A BOO-BOO IN LINE 1
```

If your computer is stuffier, it will say:

```
SYNTAX ERROR IN 1
```

If your computer is curt, it will say:

```
SN. ERROR IN 1
```

If your computer is long-winded, it will say:

```
KEYWORD NOT RECOGNIZED NOR HAS THE FORM FOR AN IMPLIED LET--CHECK FOR MISSPELLING
IN LINE 1
```

Anyway, regardless of how awful your computer's personality is, your next job is to correct your error. To do that, just retype line 1 correctly, and then type the word RUN again. So altogether, the conversation looks like this:

You type this, but it contains an error→→
```
1 PRIMT "ROSES ARE RED"
2 PRINT "CABBAGE IS GREEN"
3 PRINT "MY FACE IS FUNNY"
4 PRINT "BUT YOURS IS A SCREAM"
RUN
```
The computer gripes at you→→→→→→→→→→→→→→→→
```
YOU MADE A BOO-BOO IN LINE 1
READY
```
You type this, to correct the error→→→→→→
```
1 PRINT "ROSES ARE RED"
RUN
```
The computer recites the poem→→→→→→→→→→→→→
```
ROSES ARE RED
CABBAGE IS GREEN
MY FACE IS FUNNY
BUT YOURS IS A SCREAM
```

Exception: on Radio Shack computers, and on CP/M computers using MBASIC, press the RETURN key before you correct the error.

Common Bloopers

If you're like most beginners, you'll make these mistakes soon— if you haven't made them already!

You type the letter O instead of zero, or type zero instead of the letter O. Your typing looks correct to you, but the computer gripes.

You type something (such as RUN or your identification number), but you forget to press the RETURN key afterwards. So the computer keeps waiting for you to press it. Since the computer isn't replying, and since you don't realize the computer is waiting for you, you think the computer is broken.

You tell the computer to PRINT a message, but you forget to put the message inside quotation marks. So the computer gripes, or prints a zero instead.

You start typing a new program, but forget to type the word NEW. So when you type RUN, the computer runs a mish-mash of the new program with the previous program, and reprints some messages from the previous program.

Fun

Spaces

```
1 PRINT "JOY"
2 PRINT
3 PRINT "SORROW"
```

Line 1 makes the computer print JOY. Line 2 makes the computer print a blank line, underneath JOY. Altogether, the computer will print:

```
JOY

SORROW
```

Suppose you want the computer to print this letter:

```
DEAR JOAN,
  THANK YOU FOR THE BEAUTIFUL
NECKTIE.  JUST ONE PROBLEM--
I HATE NECKTIES!
              LOVE,
              FRED-THE-HIPPIE
```

This program prints it:

```
1 PRINT "DEAR JOAN,"
2 PRINT "  THANK YOU FOR THE BEAUTIFUL"
3 PRINT "NECKTIE.  JUST ONE PROBLEM--"
4 PRINT "I HATE NECKTIES!"
5 PRINT "              LOVE,"
6 PRINT "              FRED-THE-HIPPIE"
```

In the program, each line must contain two quotation marks. To make the computer indent a line, put blank spaces *after* the first quotation mark.

Semicolons

```
1 PRINT "FAT";
2 PRINT "HER"
```

Line 1 makes the computer print FAT; and line 1 ends with a semicolon. The semicolon makes the computer print the next item on the same line; so the computer will print HER on the same line, like this:

```
FATHER
```

This program shows what happens to an evil king on a boat:

```
1 PRINT "SIN";"KING"
```

The computer will print SIN, and will print KING on the same line, like this:

```
SINKING
```

Rearranging Your Program

You don't have to type your program in order. For example, suppose you type:

```
2 PRINT "DRINKS"
1 PRINT "MOMMY"
4 PRINT "BLOOD"
3 PRINT "MY"
```

In its mind, the computer automatically rearranges the program, so the numbers are in increasing order, like this:

```
1 PRINT "MOMMY"
2 PRINT "DRINKS"
3 PRINT "MY"
4 PRINT "BLOOD"
```

To find out what's in the computer's mind, type the word LIST. Example:

You type this program→→→→→→→→→→→→→→→→→→→→→→→→→→→→→→→2 PRINT "DRINKS"

```
1 PRINT "MOMMY"
```

```
4 PRINT "BLOOD"
3 PRINT "MY"
```

You type this command→→→→→→→→→→→→→→→→→→→→→→→→→→→→→→LIST

The computer types the program in increasing order→→1 PRINT "MOMMY"

```
2 PRINT "DRINKS"
3 PRINT "MY"
4 PRINT "BLOOD"
```

If you type RUN, the computer will print:

```
MOMMY
DRINKS
MY
BLOOD
```

Whenever you're confused, type the word LIST. Then the computer will tell you what's in its mind; and what's in its mind might surprise you!

Kissing Feet

Type this, and see what happens:

```
NEW
10 PRINT "KISS"
90 PRINT "AND CRY"
32 PRINT "SNEEZE"
50 PRINT "FEET"
32 PRINT "PRETTY"
70
80 PRINT "HAT"
80
20 PRINT "MY"
```

Since you typed two versions of line 32, the computer assumes the second is a correction. So the computer assumes that line 32 says PRINT "PRETTY".

Since you typed nothing in line 70, the computer ignores line 70.

Since you typed two versions of line 80, the computer assumes the second is a correction. Since the second says nothing, the computer assumes you want line 80 erased.

In its mind, the computer rearranges the program, so the numbers are in increasing order. If you type LIST, the computer will type:

```
10 PRINT "KISS"
20 PRINT "MY"
32 PRINT "PRETTY"
50 PRINT "FEET"
90 PRINT "AND CRY"
```

If you type RUN, the computer will print:

```
KISS
MY
PRETTY
FEET
AND CRY
```

If you think about that example, you'll notice two things:

To revise a line, just retype it.

To erase line 80, just type—
80
with nothing else on the line.

The word NEW erases your program from the computer's mind, to make way for a new one. But the word RUN does *not* erase the program; after the run, you can continue inserting, deleting, and revising lines.

Use Tens

If your first line is numbered 1, and your second line is numbered 2, you can't squeeze a line between them, since decimals aren't allowed. So expert programmers number their lines 10, 20, 30, 40,... instead of 1, 2, 3,...

GO TO

This program makes the computer go crazy:

```
10 PRINT "CAT"
20 PRINT "DOG"
30 GO TO 10
```

Line 10 makes the computer print CAT. Line 20 makes it print DOG. Line 30 makes it go back to line 10 again, so it prints CAT again. Then it prints DOG again, and comes to line 30 again, which makes it go back to line 10 again, print CAT and DOG again, and then jump back again....The computer will print the words CAT and DOG again and again, like this:

```
CAT
DOG
CAT
DOG
CAT
DOG
CAT
DOG
CAT
DOG
etc.
```

The computer will try to print the words CAT and DOG again and again, *forever*. Try running that program; you'll have fun watching the computer go crazy. If your terminal uses paper, you'll see reams of paper spewing out; so the whole floor will be buried under piles of paper that say CAT and DOG thousands of times. If your terminal uses a screen instead, the computer will print the words CAT and DOG on it faster than you can read; you'll see a blur, as thousands of CATs and DOGs go flying up the screen.

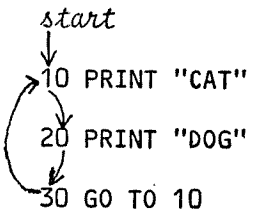
To stop this madness, you must give the computer some kind of "jolt" that will put it out of its misery. And now we come to the controversial part of this book. To put the computer out of its misery, you must give it an *abortion*. Here's how:

New-style computers	How to give an abortion
Radio Shack, HP-2000, Honeywell	Press the BREAK key.
Nova, Eclipse	Press the ESCAPE key.
PET, CBM	Press the STOP key.
Apple 2-plus	Press the RESET key.

Old-style computers	How to give an abortion
CDC	Type the word STOP. Then press the RETURN key.
PDP-11	While you hold down the CONTROL key, tap the C key.
PDP-10, PDP-20	While you hold down the CONTROL key, tap the C key *twice*.
Apple 2	While you hold down the CONTROL key, tap the C key. Then press the RETURN key.

When you give the abortion, the computer will stop its labor pains; it will *abort your program* and say READY. Then you can give any command, such as LIST.

Here's a picture of that program:



The computer follows the arrows, which make it go round and round in a *loop*. Since the computer will try to go round and round the loop forever, the loop is called *infinite*. The only way to stop an infinite loop is to give an abortion.

In that program, you typed just three lines; but since the third line said to GO TO the beginning, the computer does an infinite loop. By saying GO TO, you can make the computer do an infinite amount of work. Moral: *the computer can turn a finite amount of human energy into an infinite amount of good.* Putting it another way: *the computer can multiply your abilities by infinity.*

For example, suppose you want to send this poem to all your friends:

```
I'M HAVING TROUBLE
WITH MY NOSE.
THE ONLY THING IT DOES IS:
BLOWS!
```

Just run this program:

```
10 PRINT "I'M HAVING TROUBLE"
20 PRINT "WITH MY NOSE."
30 PRINT "THE ONLY THING IT DOES IS:"
40 PRINT "BLOWS!"
50 PRINT
60 GO TO 10
```

Lines 10-40 print the poem. Line 50 prints a blank line at the end of the poem. Line 60 makes the computer do all that repeatedly; the computer will print:

```
I'M HAVING TROUBLE
WITH MY NOSE.
THE ONLY THING IT DOES IS:
BLOWS!

I'M HAVING TROUBLE
WITH MY NOSE.
THE ONLY THING IT DOES IS:
BLOWS!

I'M HAVING TROUBLE
WITH MY NOSE.
THE ONLY THING IT DOES IS:
BLOWS!
```

etc.

The computer will print infinitely many copies— unless you give it an abortion.

Old-Fashioned Computers

On old-fashioned computers (such as Radio Shack computers having only Level-1 BASIC), you must omit the space after the word GO. Instead of saying—

```
GO TO
```

you must say:

```
GOTO
```

Love

This program's for lovers:

```
10 PRINT "LOVE"
20 GO TO 10
```

The computer will print LOVE repeatedly, like this:

```
LOVE
LOVE
LOVE
LOVE
LOVE
```
etc.

To have more lovely fun, put a semicolon at the end of line 10, like this:

```
10 PRINT "LOVE";
20 GO TO 10
```

The semicolon makes the computer print LOVE *next to* LOVE:

```
LOVELOVELOVELOVELOVELOVELOVELOVELOVELOVELOVELOVELOVELOVELOVELOVELOVELOVELOVE
LOVELOVELOVELOVELOVELOVELOVELOVELOVELOVELOVELOVELOVELOVELOVELOVELOVELOVELOVE
LOVELOVELOVELOVELOVELOVELOVELOVELOVELOVELOVELOVELOVELOVELOVELOVELOVELOVELOVE
```
etc.

If you put a space after LOVE, like this—

```
10 PRINT "LOVE ";
20 GO TO 10
```

the computer will print a space after each LOVE:

```
LOVE LOVE LOVE LOVE LOVE LOVE LOVE LOVE LOVE LOVE LOVE LOVE LOVE LOVE LOVE LOVE
LOVE LOVE LOVE LOVE LOVE LOVE LOVE LOVE LOVE LOVE LOVE LOVE LOVE LOVE LOVE LOVE
LOVE LOVE LOVE LOVE LOVE LOVE LOVE LOVE LOVE LOVE LOVE LOVE LOVE LOVE LOVE LOVE
```
etc.

Instead of making the computer print LOVE, try making it print an advertisement (such as BUY JOE'S MEAT), or a political slogan (such as STOP POLLUTION), or your name, or something else that makes you feel emotional. Computers, like dogs, imitate the personalities of their masters. If your computer acts "cold and heartless", it's because *you* are.

Computer Torture

This program tortures the computer:

```
10 GO TO 10
```

Line 10 tells the computer to go to line 10, which tells the computer to go to line 10, which tells the computer to go to line 10.... The computer will do line 10 again and again, forever. In other words, the computer will spend its entire life just mumbling to itself. That program has turned your beautiful computer into a bumbling idiot. If you're mean enough to give your computer that program, please give it an abortion! Otherwise, a member of the Association for Prevention of Cruelty to Computers will shoot you.

Human Torture

A program that makes the computer do the same thing again and again forever is an infinite loop. Some humans act just like computers. These humans do the same

thing again and again, every day. Their lives are sheer drudgery. They are caught in an infinite loop. Their only escape is for some nice guy to come along and jerk them out of their misery— by giving them an abortion.

(Somewhere in that paragraph lurks a sinister comment about our society. Is it that "abortion is good"? Is it that "everybody needs to get jerked"? Is it that "when you tell a human where to go, you're asking for trouble"? Or is it that "the ultimate infinite trip is a bummer"? My analyst hopes to help me work this out.)

Dead Snakes

Here's a poor, innocent program:

```
10 PRINT "YOUR MOTHER"
20 PRINT "HATES ANYONE WHO"
30 PRINT "KISSES DEAD SNAKES"
```

It makes the computer print:

```
YOUR MOTHER
HATES ANYONE WHO
KISSES DEAD SNAKES
```

To make the program more diabolical, insert line 15, like this:

```
          10 PRINT "YOUR MOTHER"
New line→→15 GO TO 30
          20 PRINT "HATES ANYONE WHO"
          30 PRINT "KISSES DEAD SNAKES"
```

The computer does line 10, which prints YOUR MOTHER. Then it does line 15, which makes it skip ahead to line 30, which prints KISSES DEAD SNAKES. So altogether, the computer prints:

```
YOUR MOTHER
KISSES DEAD SNAKES
```

In that program, line 15 made the computer skip ahead to line 30; in other words, it made the computer skip over line 20.

Conveniences

If you have a microcomputer or a PDP-11, PDP-20, Nova, or Eclipse, you can say PRINT without putting a line number in front:

```
PRINT "I LOVE YOU"
```

When you press the RETURN key at the end of that line, the computer will immediately print "I LOVE YOU". You don't have to type the word RUN. That short-cut is called *immediate mode*.

On most of those computers, you can short-cut the process even further; instead of typing the word PRINT, you can type this symbol:

Computer	Symbol for PRINT	How to say PRINT "I LOVE YOU"
Radio Shack Level-1	P.	P. "I LOVE YOU"
Radio Shack Level-2	?	? "I LOVE YOU"
most other microcomputers	?	? "I LOVE YOU"
PDP-11 with BASIC-PLUS	&	& "I LOVE YOU"
Nova, Eclipse	;	; "I LOVE YOU"

Radio Shack's computer prints on a television screen, usually. If you want that computer to print on paper instead, attach the computer to a printer (which prints on paper), and say LPRINT (instead of PRINT), and say LLIST (instead of LIST).

Leaving the Computer

When you're done using the computer and want to walk away, here's what to do....

If you've been using a microcomputer. Turn everything off. (The easiest way to do that is to pull out the plug.) Make sure *everything* is off.

If you've been using a minicomputer or maxicomputer. Type the word BYE, and then press the RETURN key. That tells the computer to stop worrying about you. If the computer says CONFIRM, type the letter F and then press the RETURN key. After a few seconds, the computer will print a good-bye message to you. The message will probably say that you've *signed off* or *logged off* or *logged out*— which means that you've gone away. The message will also say how many seconds the computer spent thinking about you during the session. If you were using a phone, hang up, and turn off the beeper. If your terminal used paper, turn off the terminal. If your terminal used a screen instead of paper, you should probably leave the terminal on (since somebody else will probably be using it soon, and since the screen uses little electricity); ask the people in your computer center.

Math

Arithmetic

Let's try to make the computer print the answer to 2+2. (And let's hope the computer says 4.) Run this program:

```
10 PRINT 2+2
```

(To run that program on an old-fashioned computer, you must type the word NEW, then type the program, then type the word RUN. But if you have a microcomputer or a PDP-11, PDP-20, Nova, or Eclipse, you can run that program more quickly and easily, by using immediate mode, and by using a symbol instead of the word PRINT.) The computer will print:

```
 4
```

If you want to subtract 2 from 9, run this program:

```
10 PRINT 9-2
```

The computer will print:

```
 7
```

You can use decimal points and negative numbers. For example, if you run this program—

```
10 PRINT -26.3+1
```

the computer will print:

```
-25.3
```

(To use decimals on an Apple computer, you must use Applesoft BASIC, *not* Integer BASIC.)

To multiply, use an asterisk. To multiply 2 by 6, run this program:

```
10 PRINT 2*6
```

The computer will print:

```
 12
```

To divide, use a slash. So to divide 8 by 4, run this program:

```
10 PRINT 8/4
```

The computer will print:

```
 2
```

Do *not* put commas in large numbers. To write four million, do *not* write 4,000,000; instead, write 4000000.

E Notation

If the computer types a number with an E, the E means: move the decimal point.

For example, suppose the computer says the answer to a problem is:

```
8.51673E9
```

The E means: move the decimal point. In fact, the E9 means: move the decimal point 9 places to the right. So look at 8.51673, and move the decimal point 9 places to the right; you get 8516730000. So when the computer says the answer is 8.51673E9, the computer really means the answer is 8516730000, approximately. (The exact answer might be 8516730000.2, or 8516730001.79, or some similar number; what the computer tells you is just an approximation.)

Instead of saying E9, some computers say E+9, or E+09. They mean the same thing.

Suppose your computer says the answer to a problem is:

```
9.26E-4
```

After the E, the minus sign means: move the decimal point to the *left*. So the E-4 means: move the decimal point 4 places to the *left*. So look at 9.26, and move the decimal point 4 places to the left; you get:
.000926
So when the computer says the answer is 9.26E-4, the computer really means the answer is:
.000926

You won't have to worry much about that E notation: the computer uses it only if an answer is very, very large or very, very teensy-weensy. (A typical computer uses E notation only if the answer is bigger than 999999 or tinier than .01.) But when the computer *does* use E notation, remember to move the decimal point!

Range

The highest number a typical computer can handle is about 1E38 (which is 1 followed by 38 zeros, which is 100000000000000000000000000000000000000). If you try to go much higher, the computer will say OVERFLOW ERROR.

The tiniest decimal the computer can handle is about 1E-38 (which is a decimal point followed by 38 digits, 37 of which are zeros, like this:
.00000000000000000000000000000000000001). If you try to go tinier, the computer will have an UNDERFLOW ERROR and will "fake" the answer: it will give zero instead of the correct answer.

That "safe range", from 1E-38 to 1E38, works on most computers. On some computers, the safe range is different:

Computer	Safe range
PET,CBM	1E-34 to 1E34
IBM, Nova, Eclipse	1E-78 to 1E75
CDC	1E-293 to 1E322
PDP-8	1E-616 to 1E616

Order of Operations

What does "2 plus 3 times 4" mean? The answer depends on who you ask.

To a businessman, it means "start with 2 plus 3, then multiply by 4"; that makes 5 times 4, which is 20. But to a scientist, "2 plus 3 times 4" means something different: it means "2 plus three fours", which is 2 + 4 + 4 + 4, which is 14.

Since computers were invented by scientists, computers think like scientists. If you run this program—

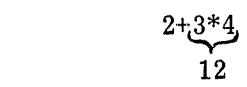
```
10 PRINT 2+3*4
```

the computer will think you mean "2 plus three fours", so it will do 2 + 4 + 4 + 4, and will print this answer:

```
 14
```

The computer will *not* print the businessman's answer, which is 20. So if you're a businessman, tough luck!

Scientists and computers follow this rule: do multiplication and division before addition and subtraction. So when your computer sees 2+3*4, it begins by hunting for multiplication and division. It finds the multiplication sign between the 3 and the 4, so it multiplies the 3 by the 4, and gets 12, like this:

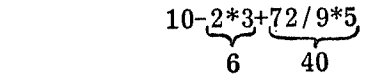


So the problem becomes 2+12, which is 14, which is the answer the computer prints.

For another example, suppose you run this program:

```
10 PRINT 10-2*3+72/9*5
```

The computer begins by doing all the multiplications and divisions. So it does 2*3 (which is 6), and does 72/9*5 (which is 8*5, which is 40), like this:



So the problem becomes 10-6+40, which is 44, which is the answer the computer prints.

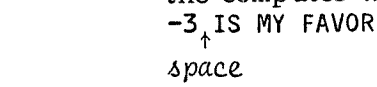
Parentheses

You can use parentheses the same way as in algebra. For example, 5-(1+1) means 5-2, which is 3. You can put parentheses inside parentheses: 10-(5-(1+1)) means 10-(5-2), which is 10-3, which is 7.

Spaces Near Numbers

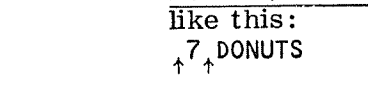
```
10 PRINT -3;"IS MY FAVORITE NUMBER"
```

Whenever the computer prints a number, it prints a blank space afterwards; so the computer will print a blank space after the -3, like this:



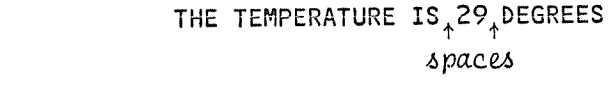

```
10 PRINT 7;"DO";"NUTS"
```

The computer prints 7 and DO and NUTS. Since 7 is a number, the computer prints a blank space after the 7. The computer prints another blank space *before* every number that's positive; so the computer prints another blank space before the 7, like this:




```
10 PRINT "THE TEMPERATURE IS";4+25;"DEGREES"
```

The computer prints THE TEMPERATURE IS, then 4+25 (which is 29), then DEGREES. Since 29 is a positive number, the computer prints a blank space before and after the 29:




```
10 PRINT "THE TEMPERATURE IS";4-25;"DEGREES"
```

The computer prints THE TEMPERATURE IS, then 4-25 (which is -21), then DEGREES. Since -21 is a number, the computer prints a space after it; but since -21 is *not* positive, the computer does *not* print a space before it. The computer prints:

That looks ugly; it would look prettier if there were a space before the -21. To insert a space, put the space inside quotation marks:

inserted space, before the quotation mark

```
10 PRINT "THE TEMPERATURE IS ";4-25;"DEGREES"
```

Then the computer will print:

```
THE TEMPERATURE IS -21 DEGREES
```

inserted space

By using semicolons, you can make the computer do many numerical calculations at once. For example, suppose you want the computer to do 6+2, and also 6-2, and also 6*2, and also 6/2. This program makes the computer do all those calculations at once:

```
10 PRINT 6+2;6-2;6*2;6/2
```

The computer prints the four answers:

```
 8  4  12  3
```

The computer prints spaces between the answers, because the computer prints a space after every number (and an additional space before every number that's positive).

Lesson 2

Make Your Computer Think

Variables

Numeric Variables

A letter can stand for a number. For example, X can stand for the number 47, as in this program:

```
10 X=47
20 PRINT X+2
```

Line 10 says X stands for the number 47. In other words, X is a name for the number 47. Line 20 says to print X+2; since X is 47, X+2 is 49; so the computer will print 49. That's the only number the computer will print; it will not print 47.

A letter that stands for a number is called a *numeric variable*. In that program, X is a numeric variable; it stands for the number 47. The *value* of X is 47. Line 10 is called an *assignment statement*, because it *assigns* 47 to X.

On Honeywell and PDP-8 computers, you must type the word LET at the beginning of each assignment statement, like this:

```
10 LET X=47
```

Another example:

```
10 Y=38
20 PRINT Y-2
```

Line 10 says Y is a numeric variable that stands for the number 38. Line 20 says to print Y-2; since Y is 38, Y-2 is 36; so the computer will print 36.

Example:

```
10 B=8
20 PRINT B*3
```

Line 10 says B is 8. Line 20 says to print B*3, which is 8*3, which is 24; so the computer will print 24.

One variable can define another:

```
10 M=6
20 P=M+1
30 PRINT M*P
```

Line 10 says M is 6. Line 20 says P is M+1, which is 6+1, which is 7; so P is 7. Line 30 says to print M*P, which is 6*7, which is 42; so the computer will print 42.

A value can change:

```
10 F=4
20 F=9
30 PRINT F*2
```

Line 10 says F's value is 4. Line 20 changes F's value to 9, so line 30 prints 18.

On the left side of the equal sign, you must have *one* variable:

Allowed	Not allowed	Not allowed
P=M+1	P-M=1	1=P-M
↑	↑ ↑	↑
one variable	*two variables*	*not a variable*

The variable on the left side of the equation is the only one that changes. For example, the statement P=M+1 changes the value of P but not M. The statement A=B changes the value of A but not B:

```
10 A=1
20 B=7
30 A=B
40 PRINT A+B
```

Line 30 changes A, to make it equal B; so A becomes 7. Since both A and B are now 7, line 40 prints 14.

"A=B" has a different effect than "B=A". That's because "A=B" changes the value of A (but not B); "B=A" changes the value of B (but not A). Compare these programs:

```
10 A=1          10 A=1
20 B=7          20 B=7
30 A=B          30 B=A
40 PRINT A+B    40 PRINT A+B
```

In the left program (which you saw before), line 30 changes A to 7, so both A and B are 7; line 40 prints 14. In the right program, line 30 changes B to 1, so both A and B are 1; line 40 prints 2.

If you don't define a numeric variable, the computer assumes it's zero:

```
10 PRINT R
```

Since R hasn't been defined, line 10 prints zero.

The computer doesn't look ahead:

```
10 PRINT J
20 J=5
```

When the computer encounters line 10, it doesn't look ahead to find out what J is. As of line 10, J is still undefined, so the computer prints zero.

When to Use Variables

When should you use variables? Here's a practical example....

Suppose you're selling something that costs $1297.43, and you want to do these calculations:

multiply $1297.43 by 2
multiply $1297.43 by .05
add $1297.43 to $483.19
divide $1297.43 by 37
subtract $1297.43 from $8598.61
multiply $1297.43 by 28.7

To do those six calculations, you could run this program:

```
10 PRINT 1297.43*2; 1297.43*.05; 1297.43+483.19; 1297.43/37; 8598.61-1297.43
20 PRINT 1297.43*28.7
```

But that program's silly, since it contains the number 1297.43 six times. This program's briefer, because it uses a variable:

```
10 C=1297.43
20 PRINT C*2; C*.05; C+483.19; C/37; 8598.61-C; C*28.7
```

So whenever you need to use a number several times, turn the number into a variable; it will make your program briefer.

String Variables

A *string* is any collection of characters, such as "I LOVE YOU" or "76 TROMBONES" or "GO AWAY!!!" or "XYPW EXR///746". Each string must be in quotation marks.

A letter can stand for a string— if you put a dollar sign after the letter. Example:

```
10 G$="DOWN"
20 PRINT G$
```

Line 10 says G$ stands for the string "DOWN". Line 20 prints:

```
DOWN
```

In that program, G$ is a variable. Since it stands for a string, it's called a *string variable*. Every string variable must end with a dollar sign; the dollar sign is supposed to remind you of a fancy S, which stands for String. Line 10 is pronounced, "G String is DOWN".

If you're paranoid, you'll love this program:

```
10 L$="THEY'RE LAUGHING AT YOU"
20 PRINT L$
30 PRINT L$
40 PRINT L$
```

Line 10 says L$ stands for the string "THEY'RE LAUGHING AT YOU". Lines 20, 30, and 40 make the computer print:

```
THEY'RE LAUGHING AT YOU
THEY'RE LAUGHING AT YOU
THEY'RE LAUGHING AT YOU
```

The computer can recite nursery rhymes:

```
10 H$="HICKORY, DICKORY, DOCK!"
20 M$="THE MOUSE"
30 C$="THE CLOCK"
40 PRINT H$
50 PRINT M$;" RAN UP ";C$
60 PRINT C$;" STRUCK ONE"
70 PRINT M$;" RAN DOWN"
80 PRINT H$
```

Lines 10-30 define H$, M$, and C$. Lines 40-80 make the computer print:

```
HICKORY, DICKORY, DOCK!
THE MOUSE RAN UP THE CLOCK
THE CLOCK STRUCK ONE
THE MOUSE RAN DOWN
HICKORY, DICKORY, DOCK!
```

If you don't define a string variable, the computer assumes it's blank:

```
10 PRINT F$
```

Since F$ hasn't been defined, line 10 makes the computer print a line that says nothing; the line the computer prints is blank.

Weird Computers

Radio Shack (using just Level-1 BASIC instead of Level-2 BASIC). The only string variables you can use are A$ and B$; you *cannot* use the other letters of the alphabet (such as C$ or D$ or E$). If you try to make A$ or B$ a string that's long, the computer will look at only the first 16 characters.

PDP-8. There are four kinds of BASIC on PDP-8 computers. Kind 1: you can use string variables without any hassles. Kind 2: you can't use string variables. Kind 3: you can use string variables, but the computer looks at only the first six characters of each string. Kind 4: read the next paragraph.

Nova, Eclipse, HP, Cromemco, IBM, PDP-8 (using kind-4 BASIC), Apple (using just Integer BASIC instead of Applesoft BASIC), North Star (using just North Star BASIC instead of CP/M's MBASIC). The first line of your program must say DIM, and must list each string variable that will be in your program. For example, if your program will use A$, B$, and N$, line 1 should say:

```
1 DIM A$(80), B$(80), N$(80)
```

The 80 lets each string have up to 80 characters. It reserves space in the computer's memory. If you know the strings will be shorter, use numbers less than 80, so you hog less space in the computer's memory. For example, if you know A$ won't have more than 30 characters, and B$ won't have more than 40 characters, and N$ won't have more than 5 characters, write this:

```
1 DIM A$(30), B$(40), N$(5)
```

On IBM computers, omit the parentheses; type just this:

```
1 DIM A$30, B$40, N$5
```

On PDP-8 computers, put an equal sign after each dollar sign, like this:

```
1 DIM A$=30, B$=40, N$=5
```

In the DIM statement, if you forget to mention a string, IBM computers will assume you meant to DIM it to 18 characters; PDP-8 computers will assume you meant to DIM it to 15 characters; Nova and Eclipse computers will assume you meant to DIM it to 10 characters.

Input

String Input

Humans ask questions. So if you want to turn the computer into a human, you must make it ask questions too. To make the computer ask a question, use the word INPUT. This program makes the computer ask for your name:

```
10 INPUT "WHAT IS YOUR NAME";N$
20 PRINT "I HATE ANYONE WHOSE NAME IS ";N$
```

Line 10 makes the computer ask "WHAT IS YOUR NAME?" and then wait for you to answer; your answer will be called N$. For example, if you answer JOEY, then N$ is JOEY. Line 20 makes the computer print:

```
I HATE ANYONE WHOSE NAME IS JOEY
```

Here's the whole conversation; I've underlined the parts typed by you:

You tell the computer to run→→→→→RUN
The computer asks for your name→→WHAT IS YOUR NAME? JOEY
The computer criticizes→→→→→→→→→→I HATE ANYONE WHOSE NAME IS JOEY

If you try that example, be careful! When you type line 10, which says INPUT, make sure you type the two quotation marks and the semicolon. You don't have to type a question mark; when the computer runs your program, it will automatically put a question mark at the end of the question.

If you wish, run that program again and pretend you're somebody else:

You tell the computer to run→→→→→RUN
The computer asks for your name→→WHAT IS YOUR NAME? GAIL
The computer criticizes→→→→→→→→→→I HATE ANYONE WHOSE NAME IS GAIL

Weird Computers

On some computers, you must write line 10 slightly differently. On Apple computers using Integer BASIC (instead of Applesoft BASIC), type a comma instead of a semicolon:

```
10 INPUT "WHAT IS YOUR NAME",N$
```

On Nova and Eclipse computers, type a comma instead of a semicolon, and insert a question mark:

```
10 INPUT "WHAT IS YOUR NAME?",N$
```

On IBM, CDC, Honeywell, Univac, PDP-8, PDP-10, and HP-2000 computers, the INPUT statement cannot contain quotation marks, so you must split the INPUT statement into two smaller statements: a statement that says to PRINT the question, and a statement that says to wait for the human to INPUT an answer:

```
10 PRINT "WHAT IS YOUR NAME";
11 INPUT N$
```

College Admissions

This program prints a letter, admitting you to the college of your choice:

```
10 INPUT "WHAT COLLEGE WOULD YOU LIKE TO BE ADMITTED TO";C$
20 PRINT "CONGRATULATIONS!"
30 PRINT "YOU HAVE JUST BEEN ADMITTED TO ";C$
40 PRINT "IT FITS YOUR PERSONALITY"
50 PRINT "I HOPE YOU GO TO ";C$
60 PRINT "           RESPECTFULLY YOURS"
70 PRINT "           THE DEAN OF ADMISSIONS"
```

Line 10 makes the computer ask "WHAT COLLEGE WOULD YOU LIKE TO BE ADMITTED TO?" and then wait for you to answer; your answer will be called C$. If you'd like to be admitted to HARVARD, you'll be pleased:

```
You tell the computer to run→→RUN
The computer asks→→→→→→→→→→→→→WHAT COLLEGE WOULD YOU LIKE TO BE ADMITTED TO? HARVARD
The computer admits you→→→→→→→CONGRATULATIONS!
                              YOU HAVE JUST BEEN ADMITTED TO HARVARD
                              IT FITS YOUR PERSONALITY
                              I HOPE YOU GO TO HARVARD
                                        RESPECTFULLY YOURS
                                        THE DEAN OF ADMISSIONS
```

You can choose any college you wish:

```
RUN
WHAT COLLEGE WOULD YOU LIKE TO BE ADMITTED TO? HELL
CONGRATULATIONS!
YOU HAVE JUST BEEN ADMITTED TO HELL
IT FITS YOUR PERSONALITY
I HOPE YOU GO TO HELL
           RESPECTFULLY YOURS
           THE DEAN OF ADMISSIONS
```

That program consists of three parts:

1. The computer begins by asking you a question ("What college would you like to be admitted to?"). The computer's question is called the *prompt*, because it prompts you to answer.
2. Your answer (the college's name) is called *your input*, because it's data that you're *putting into* the computer.
3. The computer's reply (the admission letter) is called the *computer's output*, because it's the final answer that the computer puts out.

Opposites

The word INPUT is the opposite of the word PRINT. The word PRINT makes the computer print information out; the word INPUT makes the computer take information in. The word PRINT makes the computer output; the word INPUT makes the computer

accept your input. Input and Output are collectively called I/O, so the INPUT and PRINT statements are called I/O statements.

Stories

Let's make the computer write a story, by filling in the blanks:

```
ONCE UPON A TIME, THERE WAS A YOUNGSTER NAMED _________
                                              your name

WHO HAD A FRIEND NAMED _____________
                       friend's name

_________ WANTED TO _____________________ _____________
your name           verb, such as "hit"   friend's name

BUT _____________ DIDN'T WANT TO _____________________ _________
    friend's name                verb, such as "hit"   your name

WILL _________ _____________________ _____________
     your name verb, such as "hit"   friend's name

WILL _____________ _____________________ _________
     friend's name verb, such as "hit"   your name

TO FIND OUT, COME BACK AND SEE THE NEXT EXCITING EPISODE

OF _________ AND _____________
   your name     friend's name
```

To write the story, the computer must ask for your name, your friend's name, and a verb. To make the computer ask, your program must say INPUT. Here's the program:

```
The computer asks→→→→10 INPUT "WHAT'S YOUR NAME";Y$
                     20 INPUT "WHAT'S YOUR FRIEND'S NAME";F$
                     30 INPUT "IN 1 WORD, SAY SOMETHING YOU CAN DO TO YOUR FRIEND";V$
The computer writes→→40 PRINT "HERE'S MY STORY...."
                     50 PRINT "ONCE UPON A TIME, THERE WAS A YOUNGSTER NAMED ";Y$
                     60 PRINT "WHO HAD A FRIEND NAMED ";F$
                     70 PRINT Y$;" WANTED TO ";V$;" ";F$
                     80 PRINT "BUT ";F$;" DIDN'T WANT TO ";V$;" ";Y$
                     90 PRINT "WILL ";Y$;" ";V$;" ";F$
                     100 PRINT "WILL ";F$;" ";V$;" ";Y$
                     110 PRINT "TO FIND OUT, COME BACK AND SEE THE NEXT EXCITING EPIS
                     ODE"
                     120 PRINT "OF ";Y$;" AND ";F$
```

Here's a sample run:

```
WHAT'S YOUR NAME? DRACULA
WHAT'S YOUR FRIEND'S NAME? MARILYN MONROE
IN 1 WORD, SAY SOMETHING YOU CAN DO TO YOUR FRIEND? BITE
HERE'S MY STORY....
ONCE UPON A TIME, THERE WAS A YOUNGSTER NAMED DRACULA
WHO HAD A FRIEND NAMED MARILYN MONROE
DRACULA WANTED TO BITE MARILYN MONROE
BUT MARILYN MONROE DIDN'T WANT TO BITE DRACULA
WILL DRACULA BITE MARILYN MONROE
WILL MARILYN MONROE BITE DRACULA
TO FIND OUT, COME BACK AND SEE THE NEXT EXCITING EPISODE
OF DRACULA AND MARILYN MONROE
```

Here's another run:

```
WHAT'S YOUR NAME? SUPERMAN
WHAT'S YOUR FRIEND'S NAME? KING KONG
IN 1 WORD, SAY SOMETHING YOU CAN DO TO YOUR FRIEND? TICKLE
HERE'S MY STORY....
ONCE UPON A TIME, THERE WAS A YOUNGSTER NAMED SUPERMAN
WHO HAD A FRIEND NAMED KING KONG
SUPERMAN WANTED TO TICKLE KING KONG
BUT KING KONG DIDN'T WANT TO TICKLE SUPERMAN
WILL SUPERMAN TICKLE KING KONG
WILL KING KONG TICKLE SUPERMAN
TO FIND OUT, COME BACK AND SEE THE NEXT EXCITING EPISODE
OF SUPERMAN AND KING KONG
```

Try it: put in your own name, the name of your friend, and something you'd like to do to your friend.

This program creates a story that's fancier:

```
10 INPUT "WHAT'S YOUR NAME";Y$
20 INPUT "WHAT'S YOUR FRIEND'S NAME";F$
30 INPUT "WHAT'S THE NAME OF ANOTHER FRIEND";A$
40 INPUT "NAME A COLOR";C$
50 INPUT "NAME A PLACE";P$
60 INPUT "NAME A FOOD";D$
70 INPUT "NAME AN OBJECT";J$
80 INPUT "NAME A PART OF THE BODY";B$
90 INPUT "NAME A STYLE OF COOKING (SUCH AS BAKED OR FRIED)";S$
100 PRINT
110 PRINT "CONGRATULATIONS ";Y$
120 PRINT "YOU'VE WON THE BEAUTY CONTEST, BECAUSE OF YOUR SEXY ";B$
130 PRINT "YOUR PRIZE IS A ";C$;" ";J$
140 PRINT "PLUS A TRIP TO ";P$;" WITH YOUR FRIEND ";F$
150 PRINT "PLUS...AND THIS IS THE BEST PART OF ALL..."
160 PRINT "DINNER FOR THE TWO OF YOU AT ";A$;"'S NEW RESTAURANT"
170 PRINT "WHERE ";A$;" WILL GIVE YOU ALL THE ";S$;" ";F$;" YOU CAN EAT"
180 PRINT "CONGRATULATIONS ";Y$;"...TODAY'S YOUR LUCKY DAY..."
190 PRINT "NOW EVERYBODY WANTS TO KISS YOUR AWARD-WINNING ";B$
```

Here's a sample run:

```
WHAT'S YOUR NAME? IVAN
WHAT'S YOUR FRIEND'S NAME? FRITZ
WHAT'S THE NAME OF ANOTHER FRIEND? BRIGITTE
NAME A COLOR? RED
NAME A PLACE? TIMES SQUARE
NAME A FOOD? CHERRIES
NAME AN OBJECT? TOILET
NAME A PART OF THE BODY? ASS
NAME A STYLE OF COOKING (SUCH AS BAKED OR FRIED)? STEAMED

CONGRATULATIONS IVAN
YOU'VE WON THE BEAUTY CONTEST, BECAUSE OF YOUR SEXY ASS
YOUR PRIZE IS A RED TOILET
PLUS A TRIP TO TIMES SQUARE WITH YOUR FRIEND FRITZ
PLUS...AND THIS IS THE BEST PART OF ALL...
DINNER FOR THE TWO OF YOU AT BRIGITTE'S NEW RESTAURANT
WHERE BRIGITTE WILL GIVE YOU ALL THE STEAMED CHERRIES YOU CAN EAT
CONGRATULATIONS IVAN...TODAY'S YOUR LUCKY DAY...
NOW EVERYBODY WANTS TO KISS YOUR AWARD-WINNING ASS
```

String Space

If your computer is a Sorcerer or a Compucolor, or uses Radio Shack's Level-2 BASIC or CP/M's MBASIC, it might gripe by saying—

```
OUT OF STRING SPACE
```

or:

```
?OS ERROR
```

To stop the computer from griping, insert this statement:

```
1 CLEAR 300
```

If the computer *still* gives that kind of gripe, raise 300 to a higher number, like this:

```
1 CLEAR 500
```

Keep raising, until the computer stops that gripe. On the other hand, if you raise *too* high, the computer will say—

```
OUT OF MEMORY
```

or:

```
?OM ERROR
```

That means you'll have to go lower.

Why You Must Clear

Here's *why* you must say CLEAR, on those computers....

Your program contains two kinds of strings: the strings that are known, and the strings that are unknown. For example, if you say—

```
10 G$="DOWN"
```

the G$ is known to be DOWN. If you say—

```
10 INPUT "WHAT IS YOUR NAME";N$
```

the N$ is unknown, because its value depends on what the human inputs.

Normally, the computer's memory reserves enough space to hold 50 unknown characters. If your program needs *more* than 50 unknown characters— for example, if your program inputs a string that contains 49 characters, plus another string that contains 2 characters— the computer will gripe, and say OUT OF STRING SPACE.

If you say—

```
1 CLEAR 300
```

the computer will reserve enough space to hold 300 unknown characters, instead of 50.

How much space can you reserve? That depends on how much memory you bought for your computer. If your computer has little memory, and you try to reserve more space than your computer has, the computer will give up, and say OUT OF MEMORY.

Bills

If you're a nasty businessman, you'll love this program:

```
10 INPUT "WHAT IS THE CUSTOMER'S FIRST NAME";F$
20 INPUT "LAST NAME";L$
30 INPUT "STREET ADDRESS";A$
40 INPUT "CITY";C$
50 INPUT "STATE";S$
60 INPUT "ZIP CODE";Z$
70 PRINT
80 PRINT F$;" ";L$
90 PRINT A$
100 PRINT C$;" ";S$;" ";Z$
```

```
110 PRINT
120 PRINT "DEAR ";F$
130 PRINT "   YOU STILL HAVEN'T PAID THE BILL."
140 PRINT "IF YOU DON'T PAY IT SOON, ";F$;","
150 PRINT "I'LL COME VISIT YOU IN ";C$
160 PRINT "AND PERSONALLY SHOOT YOU."
170 PRINT "            YOURS TRULY,"
180 PRINT "            SURE-AS-SHOOTIN'"
190 PRINT "            YOUR CRAZY CREDITOR"
```

Can you figure out what that program does?

Numeric Input

This program makes the computer predict your future:

```
10 PRINT "I WILL PREDICT WHAT WILL HAPPEN TO YOU IN THE YEAR 2000!"
20 PRINT "IN WHAT YEAR WERE YOU BORN";Y
30 PRINT "IN THE YEAR 2000, YOU WILL BECOME";2000-Y;"YEARS OLD."
```

Here's a sample run:

```
I WILL PREDICT WHAT WILL HAPPEN TO YOU IN THE YEAR 2000!
IN WHAT YEAR WERE YOU BORN? 1957
IN THE YEAR 2000, YOU WILL BECOME 43 YEARS OLD.
```

Suppose you're selling tickets to a play. Each ticket costs $2.79. (You decided $2.79 would be a nifty price, because the cast has 279 people.) This program finds the price of multiple tickets:

```
10 INPUT "HOW MANY TICKETS";T
20 PRINT "THE TOTAL PRICE IS $";T*2.79
```

This program tells you how much the "energy crisis" costs you, when you drive your car:

```
10 INPUT "HOW MANY MILES DO YOU WANT TO DRIVE";M
20 INPUT "HOW MANY PENNIES DOES A GALLON OF GAS COST";P
30 INPUT "HOW MANY MILES-PER-GALLON DOES YOUR CAR GET";R
40 PRINT "THE GAS FOR YOUR TRIP WILL COST YOU $";M*P/(R*100)
```

Here's a sample run:

```
HOW MANY MILES DO YOU WANT TO DRIVE? 300
HOW MANY PENNIES DOES A GALLON OF GAS COST? 97.9
HOW MANY MILES-PER-GALLON DOES YOUR CAR GET? 27
THE GAS FOR YOUR TRIP WILL COST YOU $ 10.8778
```

Conversion

This program converts feet to inches:

```
10 INPUT "HOW MANY FEET";F
20 PRINT F;"FEET =";F*12;"INCHES"
```

Here's a sample run:

```
HOW MANY FEET? 3
 3 FEET = 36 INCHES
```

Try to convert to the metric system? This program converts inches to centimeters:

```
10 INPUT "HOW MANY INCHES";I
20 PRINT I;"INCHES =";I*2.54;"CENTIMETERS"
```

Nice day today, isn't it? This program converts the temperature from Celsius to Fahrenheit:

```
10 INPUT "HOW MANY DEGREES CELCIUS";C
20 PRINT C;"DEGREES CELCIUS =";C*1.8+32;"DEGREES FAHRENHEIT"
```

Here's a sample run:

```
HOW MANY DEGREES CELCIUS? 20
 20 DEGREES CELCIUS = 68 DEGREES FAHRENHEIT
```

See, you can write the *Guide* yourself! Just hunt through any old math or science book, find any old formula (such as F=C*1.8+32), and turn it into a program.

Auxiliary Memory

Disks

While you're working on a program, the computer keeps it in the *main memory*. It is erased from main memory when you type NEW (to begin a new program) or BYE, or when somebody turns off the computer's power.

To store the program longer, copy it onto the computer's *disks*, which look like phonograph records. Once you've copied the program onto a disk, it will remain on the disk, even if you type NEW or BYE or turn off the power. It will stay on the disk permanently.

To copy a program onto the disk, type the program and then type the word SAVE. You must also invent a name for the program.

For example, suppose you want to copy this program onto the disk:

```
10 PRINT "MY DAD"
20 PRINT "IS BAD"
```

To name that program "JOE", and copy it onto the disk, type this:

Apple
```
NEW
10 PRINT "MY DAD"
20 PRINT "IS BAD"
SAVE JOE
```

Radio Shack, Nova, Eclipse
```
NEW
10 PRINT "MY DAD"
20 PRINT "IS BAD"
SAVE "JOE"
```

PDP, Honeywell
```
NEW JOE
10 PRINT "MY DAD"
20 PRINT "IS BAD"
30 END  (maybe)
SAVE
```

CDC
```
NEW,JOE
10 PRINT "MY DAD"
20 PRINT "IS BAD"
SAVE
```

HP-2000
```
SCRATCH
NAME-JOE
10 PRINT "MY DAD"
20 PRINT "IS BAD"
30 END
SAVE
```

Instead of JOE, you can invent a different name. Here's how long the name can be:

Computers	How long the program's name can be
PDP-10, PDP-11	6 characters
CDC	7 characters
Radio Shack, Honeywell	8 characters
Nova, Eclipse (using DOS or RDOS)	10 characters
Apple	30 characters
Eclipse (using AOS)	31 characters
PDP-20	39 characters

To prove the program is on the disk, type this command:

Apple
```
CATALOG
```

Most computers
```
CAT
```

Nova, Eclipse
```
FILE
```

Radio Shack
```
CMD "S"  (means: COMMAND SYSTEM)
DIR      (means: DIRECTORY)
BASIC
```

That makes the computer print a catalog of everything you've stored on disks.

The catalog will list the names of all the programs you've stored. And one of the names you'll see will be JOE.

Suppose you come back to the computer sometime in the future, and want to use JOE, which is on the disk. Type this:

Apple	Radio Shack, Nova, Eclipse	PDP, Honeywell	CDC	HP-2000
LOAD JOE	LOAD "JOE"	OLD JOE	OLD,JOE	GET-JOE

That makes the computer copy JOE from the disk to the main memory, and say READY. Then you can type LIST or RUN, and the computer will list or run JOE.

If you ever want to erase JOE from the disk, say this:

Apple	Nova, Eclipse	Radio Shack	HP-2000	CDC	PDP, Honeywell
DELETE JOE	DELETE "JOE"	KILL "JOE"	KILL-JOE	PURGE,JOE	UNSAVE JOE

If you ever want to revise the version of JOE that's on the disk, copy JOE from the disk to the main memory (by typing LOAD JOE or LOAD "JOE" or OLD JOE or OLD,JOE or GET-JOE). Then type your revisions, by retyping some of the program's lines, or by adding new ones. Finally, say this:

Apple	Radio Shack	Nova, Eclipse	PDP-20	PDP-10, PDP-11, Honeyw., CDC	HP-2000
SAVE JOE	SAVE "JOE"	SAVE "JOE"	SAVE	REPLACE	KILL-JOE SAVE
		The computer will say "TYPE CR TO DELETE". Then press the RETURN key.			

That makes the computer replace JOE by your new version.

Tapes

Most Americans buy tape recorders, and use them to play music (such as the Beatles) or to record speeches. The most popular tape recorders use *cassette* tapes.

You can attach your microcomputer to a cassette tape recorder. On the tape, you can store your program. So the tape serves the same purpose as a disk. The tape doesn't work as well as the disk: the tape is slower and less reliable. Nevertheless, tapes are more popular than disks, because tapes are cheaper.

If you're lucky, the tape recorder is attached to your computer already. If you're unlucky, and have to attach it yourself, make sure you match the right wires with the right holes, and make sure you turn the VOLUME dial to the right number; for details, read the manual that came with your computer.

How to Copy a Program Onto a Tape

Type the program.

Put the tape into the recorder. (To do that, you might have to press the EJECT button.)

Rewind the tape, by pressing the REWIND button. (To do that on a Radio Shack computer using a CTR-41 recorder, unplug the REMOTE wire.) When the tape is rewound, press the STOP button.

If the recorder has a counter, push the counter's button, so the counter becomes zero.

Press the FAST FORWARD button, until you get to a part of the tape that's blank and good. (The very beginning of the tape is bad. Wait until the counter gets to at least 10.) Then press the STOP button.

If the recorder has a counter, notice the counter's number, and write it onto the tape's label, by using a pencil.

The rest of the procedure goes like this....

Apple. Type the word SAVE, without pressing the RETURN key. Press the RECORD and PLAY buttons, at the same time as each other. Press the RETURN key. The computer will start copying the program onto tape. The computer will give a beep.

When the computer finishes copying, it will beep again. Then it will print the symbol ">" or "]". Press the STOP button.

Radio Shack. If you had unplugged the REMOTE wire, plug it back in. Press the RECORD and PLAY buttons, at the same time as each other. Type this:

```
CSAVE "A"
```

At the end of that line, press the RETURN key. The computer will copy the program onto tape and then say READY. Press the STOP button.

PET, CBM. Type the word SAVE; then press the RETURN key. The computer will say:

```
PRESS PLAY & RECORD ON TAPE #1
```

Obey the computer, by pressing the PLAY and RECORD buttons at the same time as each other. The computer will say:

```
OK
WRITING
READY.
```

Then press the STOP button.

How to Read a Program From a Tape

Put the tape into the recorder. (To do that, you might have to press the EJECT button.) The rest of the procedure goes like this....

Apple. Press the REWIND button. When the tape is rewound, press the STOP button. Push the counter's button, so the counter becomes zero. Press the FAST FORWARD button; look at the counter; when the number on the counter is almost up to the number of the program's beginning, press the STOP button. Type the word LOAD, without pressing the RETURN key. Press the PLAY button. When the counter reaches the number of your program's beginning, press the RETURN key. The computer will start reading the tape. The computer will give a beep. When the computer has finished reading the program, it will beep again and then print the symbol ">" or "]". Press the STOP button. Type the word LIST or RUN; then press the RETURN key.

Radio Shack. Type the word CLOAD; then press the RETURN key. Press the REWIND button. When the tape is rewound, press the STOP button. Push the counter's button, so the counter becomes zero. Press the FAST FORWARD button; look at the counter; when the number on the counter is almost up to the number of the program's beginning, press the STOP button. Press the PLAY button. The computer will read the program from the tape. At the top of the screen, you'll see a pair of asterisks; the asterisk on the right should flash. (If that asterisk stays on, instead of flashing, the volume is too high; if that asterisk stays off, instead of flashing, the volume is too low. Adjust the volume dial. If one of the asterisks turns into the letter C, the tape is bad.) If all goes well, the computer will say READY. Press the STOP button. Type the word LIST or RUN; then press the RETURN key.

PET, CBM. By using the REWIND, FAST FORWARD, PLAY, and STOP buttons (and the counter, if you have one), position the tape to the program's beginning (or slightly before). Press the STOP button. Type the word LOAD; then press the RETURN key. The computer will say:

```
PRESS PLAY ON TAPE #1
```

Obey the computer, by pressing the PLAY button. The computer will say:

```
OK
SEARCHING
FOUND
LOADING
READY.
```

Press the STOP button. Type the word LIST or RUN; then press the RETURN key.

Logic

STOP

The computer understands the word STOP:

```
10 PRINT "BUBBLE GUM"
20 STOP
30 PRINT "FOX"
```

The computer will print BUBBLE GUM and then stop, without printing FOX. After the computer stops, it'll say READY. Then you can type any command, such as LIST (to see the program again) or RUN (to make the computer print BUBBLE GUM again) or NEW (to create a new program).

Colons

Microcomputers let you put several statements on the same line, like this:

```
10 A=5: B=7: PRINT A+B
```

That line says A is 5, and B is 7, and to print A+B. So the computer will print 12.

On PDP computers, type a backslash instead of a colon; type this:

```
10 A=5\ B=7\ PRINT A+B
```

IF

Let's make the computer interrogate a human. Let's make the computer begin the interrogation by asking whether the human is male or female. If the human answers MALE, let's make the computer say:

```
SO IS FRANKENSTEIN
```

If the human answers FEMALE, let's make the computer say:

```
SO IS MARY POPPINS
```

If the human gives a different answer (such as SUPER-MALE or MACHO or NOT SURE or BOTH or YES), let's make the computer say—

```
PLEASE SAY 'MALE' OR 'FEMALE'
ARE YOU MALE OR FEMALE?
```

— and force the human to answer the question correctly.

This program does it:

```
10 INPUT "ARE YOU MALE OR FEMALE";A$
20 IF A$="MALE" THEN PRINT "SO IS FRANKENSTEIN": STOP
30 IF A$="FEMALE" THEN PRINT "SO IS MARY POPPINS": STOP
40 PRINT "PLEASE SAY 'MALE' OR 'FEMALE'": GO TO 10
```

Line 10 makes the computer ask "ARE YOU MALE OR FEMALE?" and wait for the human's answer, which is called A$. If the human's answer is MALE, line 20 makes the computer print SO IS FRANKENSTEIN and then stop. If the human's answer is FEMALE, line 30 makes the computer print SO IS MARY POPPINS and then stop. If the human's answer is neither MALE nor FEMALE, the computer skips over lines 20 and 30, so it comes to line 40, which makes it print PLEASE SAY 'MALE' OR 'FEMALE' and then go back to line 10, which forces the human to answer the question again.

Here's a sample run:

```
RUN
ARE YOU MALE OR FEMALE? MALE
SO IS FRANKENSTEIN
```

Here's another:

```
RUN
ARE YOU MALE OR FEMALE? FEMALE
```

```
SO IS MARY POPPINS
```

Here's another:

```
RUN
ARE YOU MALE OR FEMALE? SUPER-MALE
PLEASE SAY 'MALE' OR 'FEMALE'
ARE YOU MALE OR FEMALE? MALE
SO IS FRANKENSTEIN
```

Let's extend the conversation. If the human says FEMALE, let's make the computer say SO IS MARY POPPINS and then ask "DO YOU LIKE HER?" If the human says YES, let's make the computer say:

```
I LIKE HER TOO--SHE IS MY MOTHER
```

If the human says NO, let's make the computer say:

```
NEITHER DO I--SHE STILL OWES ME A DIME
```

If the human says neither YES nor NO, let's make the computer say:

```
PLEASE SAY 'YES' OR 'NO'
DO YOU LIKE HER?
```

Here's the program:

```
10 INPUT "ARE YOU MALE OR FEMALE";A$
20 IF A$="MALE" THEN PRINT "SO IS FRANKENSTEIN": STOP
30 IF A$="FEMALE" THEN PRINT "SO IS MARY POPPINS": GO TO 100
40 PRINT "PLEASE SAY 'MALE' OR 'FEMALE'": GO TO 10

100 INPUT "DO YOU LIKE HER";B$
110 IF B$="YES" THEN PRINT "I LIKE HER TOO--SHE IS MY MOTHER": STOP
120 IF B$="NO" THEN PRINT "NEITHER DO I--SHE STILL OWES ME A DIME": STOP
130 PRINT "PLEASE SAY 'YES' OR 'NO'": GO TO 100
```

Line 30 says: if the human's answer is FEMALE, print SO IS MARY POPPINS and then go to line 100, which asks "DO YOU LIKE HER?"

Weird Computers

On PDP computers, replace each colon by a backslash, like this:

```
10 INPUT "ARE YOU MALE OR FEMALE";A$
20 IF A$="MALE" THEN PRINT "SO IS FRANKENSTEIN"\ STOP
30 IF A$="FEMALE" THEN PRINT "SO IS MARY POPPINS"\ GO TO 100
40 PRINT "PLEASE SAY 'MALE' OR 'FEMALE'"\ GO TO 10

100 INPUT "DO YOU LIKE HER";B$
110 IF B$="YES" THEN PRINT "I LIKE HER TOO--SHE IS MY MOTHER"\ STOP
120 IF B$="NO" THEN PRINT "NEITHER DO I--SHE STILL OWES ME A DIME"\ STOP
130 PRINT "PLEASE SAY 'YES' OR 'NO'"\ GO TO 100
```

On old-fashioned computers, such as IBM, CDC, Univac, Honeywell, Nova, Eclipse, HP-2000, Apple (without Applesoft), North Star (without CP/M), PDP-8, PDP-10 (without U. Penn BASIC), and PDP-20 (without BASIC-PLUS-2), you must replace line 20 by these shorter statements:

```
20 IF A$<>"MALE" THEN 30
21     PRINT "SO IS FRANKENSTEIN"
22     STOP
```

They say: if A$ is *not* MALE, skip ahead to line 30; otherwise, print SO IS FRANKENSTEIN and stop. Similarly, replace line 30 by these shorter statements:

```
30 IF A$<>"FEMALE" THEN 40
31     PRINT "SO IS MARY POPPINS"
```

```
32      GO TO 100
```

Also replace lines 110 and 120. This version works on practically all computers:

```
10 PRINT "ARE YOU MALE OR FEMALE";
11 INPUT A$

20 IF A$<>"MALE" THEN 30
21      PRINT "SO IS FRANKENSTEIN"
22      STOP

30 IF A$<>"FEMALE" THEN 40
31      PRINT "SO IS MARY POPPINS"
32      GO TO 100

40 PRINT "PLEASE SAY 'MALE' OR 'FEMALE'"
41 GO TO 10

100 PRINT "DO YOU LIKE HER";
101 INPUT B$

110 IF B$<>"YES" THEN 120
111      PRINT "I LIKE HER TOO--SHE IS MY MOTHER"
112      STOP

120 IF B$<>"NO" THEN 130
121      PRINT "NEITHER DO I--SHE STILL OWES ME A DIME"
122      STOP

130 PRINT "PLEASE SAY 'YES' OR 'NO'"
131 GO TO 100
```

Strange Programs

The computer is like a human; it would like to make new friends. This program makes the computer show its true feelings:

```
10 INPUT "ARE YOU MY FRIEND";A$
20 IF A$="YES" THEN PRINT "THAT'S SWELL": STOP
30 IF A$="NO" THEN PRINT "GO JUMP IN A LAKE": STOP
40 PRINT "PLEASE SAY 'YES' OR 'NO'": GO TO 10
```

When you type RUN, the computer asks "ARE YOU MY FRIEND?" If you say YES, the computer says THAT'S SWELL. If you say NO, the computer says GO JUMP IN A LAKE.

The most inventive programmers are kids. This program was written by a girl in the sixth grade:

```
10 INPUT "CAN I COME OVER TO YOUR HOUSE TO WATCH T.V.";A$
20 IF A$="YES" THEN PRINT "THANKS.  I'LL BE THERE AT 5 P.M.": STOP
30 IF A$="NO" THEN PRINT "HUMPH!  YOUR FEET SMELL, ANYWAY.": STOP
40 PRINT "PLEASE SAY 'YES' OR 'NO'": GO TO 10
```

When you type RUN, the computer asks to watch your television. If you say YES, the computer promises to come to your house at 5. If you refuse, the computer insults your feet.

Another sixth-grade girl wrote this program, to test your honesty:

```
10 PRINT "FKGJDFGKJ*#K$JSLF*%#$()$&(IKJNHBGD52:?./KSDJK$E(EF$#%JIK(*"
20 PRINT "FASDFJKL:JFRFVFJUNJI*&()JNE$#SKI#(!SERF HHW NNWAZ MAME !!!"
30 PRINT "ZBB%%%%%##)))))FESDFJK DSFE N.D.JJUJASD EHWLKD******"
40 INPUT "DO YOU UNDERSTAND WHAT I SAID";A$
50 IF A$="NO" THEN PRINT "SORRY TO HAVE BOTHERED YOU": STOP
60 IF A$="YES" THEN GO TO 100
70 PRINT "PLEASE SAY 'YES' OR 'NO'": GO TO 10

100 PRINT "SSSFJSLFKDJFL++++4567345677839XSDWFEGR%#$&&*()---==!!ZZXX"
```

```
110 PRINT "###EDFHTG NVFDF MKJKF ==+--*$&% #RHFS SESD DOPEKKKNJGFD DSBS"
120 INPUT "OKAY, WHAT DID I SAY";B$
130 PRINT "YOU ARE A LIAR, A LIAR, A BIG FAT LIAR!"
```

When you type RUN, lines 10-30 print nonsense. Then the computer asks whether you understand that stuff. *If you're honest* and answer NO, the computer will apologize. But *if you pretend that you understand the nonsense*, and answer YES, the computer will print more nonsense, challenge you to translate it, wait for you to fake a translation, and then scold you for lying.

A Daddy wrote a program for his five-year-old son, John. When John types RUN, the computer asks "WHAT'S 2 AND 2?" If John answers 4, the computer says NO, 2 AND 2 IS 22. If he runs the program again and answers 22, the computer says NO, 2 AND 2 IS 4. No matter how many times he runs the program and how he answers the question, the computer says he's wrong. But when Daddy runs the program, the computer replies, YES, DADDY IS ALWAYS RIGHT. Here's how Daddy programmed the computer:

```
10 INPUT "WHAT'S YOUR NAME";N$
20 INPUT "WHAT'S 2 AND 2";A
30 IF N$="DADDY" THEN PRINT "YES, DADDY IS ALWAYS RIGHT": STOP
40 IF A=4 THEN PRINT "NO, 2 AND 2 IS 22": STOP
50 PRINT "NO, 2 AND 2 IS 4"
```

Your Personality

This program makes the computer act human:

```
10 PRINT "HELLO"
20 INPUT "WHAT'S YOUR NAME";N$
30 PRINT "GLAD TO MEET YOU, ";N$
40 INPUT "HOW ARE YOU FEELING TODAY";F$
50 IF F$="FINE" THEN PRINT "THAT'S GOOD": STOP
60 IF F$="LOUSY" THEN PRINT "TOO BAD": STOP
70 PRINT "I FEEL THE SAME WAY"
```

When you type RUN, the computer begins the conversation by saying HELLO and asking for your name. Then it says GLAD TO MEET YOU, followed by your name; for example, if you said your name was JOEY, the computer will say:

```
GLAD TO MEET YOU, JOEY
```

Then the computer asks how you're feeling. If you say FINE, line 50 makes the computer say THAT'S GOOD. If you say LOUSY, line 60 makes the computer say TOO BAD. If you say neither FINE nor LOUSY, the computer skips lines 50 and 60 and arrives at line 70, so it says I FEEL THE SAME WAY (which is a safe, general response).

That program makes the computer imitate an "average American". But *you're not average!* Make the computer imitate *your* personality!

The first line of your program should say—

```
10 PRINT "HELLO"
```

or—

```
10 PRINT "HI"
```

or— if you're from Texas—

```
10 PRINT "HOWDY, PARDNER!"
```

or— if you're a cool jive cat—

```
10 PRINT "HI, BABY!  WHAT'S HAPPENING?"
```

or— if you're a nurse working in a private practice—

```
10 PRINT "GOOD MORNING.  THE DOCTOR WILL BE WITH YOU SHORTLY."
```

or— if you're a jerk—

```
10 PRINT "BUG OFF, OR I'LL CUT YOU UP!"
```

After your opening line, you'll need an INPUT statement, that makes the computer ask the human a question. After the INPUT statement, you'll need a series of IF statements, that make the computer react to the various responses the human might give.

Try to make your program long, so the conversation between the computer and human lasts *several minutes*. Your program will be a mass of PRINT, INPUT, IF, and GO TO statements, plus a few STOPs.

See how human you can make your computer! See how well you can make the computer imitate *you*! And remember to SAVE your program, so the computer will adopt your personality *permanently*.

Fancy Clauses

You can make the IF clause very fancy:

IF clause	Meaning
`IF A$="MALE"`	If A$ is "MALE"
`IF A=4`	If A is 4
`IF A<4`	If A is less than 4
`IF A<=4`	If A is less than or equal to 4
`IF A>4`	If A is greater than or equal to 4
`IF A>=4`	If A is greater than or equal to 4
`IF A<>4`	If A is not 4
`IF A$<"MALE"`	If A$ is a word that comes before "MALE" in the dictionary
`IF A$>"MALE"`	If A$ is a word that comes after "MALE" in the dictionary

Lesson 3

Master Your Computer

Helpful Hints

Reminders

Whenever you're confused, type this command:

```
LIST
```

You must put quotation marks around each string, and a dollar sign after each string variable:

Right	Wrong	Wrong
A$="JERK"	A$=JERK	A="JERK"

To make the computer do your program again and again, automatically, add an extra line at the bottom of your program. That extra line should tell the computer to GO TO the top line.

Here are two reasons why the computer might print too much:

You forgot to insert the word STOP into your program.
You forgot to type NEW; so the computer is reprinting part of the previous program.

To say "not equal to", say "less than or greater than", like this: <>.

Here's the "proper" way to write an IF statement....The IF statement should contain the word THEN. Do *not* put a comma before THEN. In an IF clause, the equal sign should come last:

Right	Wrong
<=	=<
>=	=>

The only way to become a computer expert is to try writing your own programs.

Debugging

If you write your own program and run it, probably it won't work. Your first reaction will be to blame the computer; don't! The probability is 99.99% that the fault is yours. Your program contains an error. An error is called a *bug*. Your next task is to *debug* the program, which means get the bugs out.

Bugs are common; top-notch programmers make errors all the time. If you write a program that works perfectly on the first run and doesn't need debugging, it's called a *gold-star program*, and means you should have tried writing a harder one instead!

It's easy to write a program that's almost completely correct, but hard to find the little bug that's fouling it up. Most of the time you spend in the computer center will be devoted to debugging.

Debugging can be fun. Hunting for the bug is like going on a treasure hunt—or solving a murder mystery. Pretend you're Sherlock Holmes. Your mission: to find the bug, and squash it!

To find the bug, use three techniques:

Inspect the program.
Trace the computer's thinking.
Shorten the program.

Here are the details....

Inspect the Program

Take a good, hard look at the program. If you stare hard enough, maybe you'll see the bug.

Ask the computer to help you. Make the computer print the entire program. To do that, type:

```
LIST
```

Usually, the bug will turn out to be just a typing error. Maybe you:

typed the letter O instead of zero? or typed zero instead of the letter O?
typed I instead of 1? or typed 1 instead of I?
pressed the SHIFT key when you weren't supposed to? or *forgot* to press it?
typed an extra letter? or *omitted* a letter?
typed a line you thought you hadn't? or *omitted* a line?

Trace the Computer's Thinking

If you've inspected the program thoroughly, and *still* haven't found the bug, the next step is to *trace* the computer's thinking. Pretend *you* are the computer. Do what your program says. Do you find yourself printing the same wrong answers the computer printed? If so, why?

To help your analysis, make the computer print *everything* it's thinking, while it's running your program. For example, suppose your program contains lines 10, 20, 30, and 40, and uses the variables B, C, and X$. Insert these lines into your program:

```
15 PRINT "I'M AT LINE 15.  THE VALUES ARE";B;C;X$
25 PRINT "I'M AT LINE 25.  THE VALUES ARE";B;C;X$
35 PRINT "I'M AT LINE 35.  THE VALUES ARE";B;C;X$
45 PRINT "I'M AT LINE 45.  THE VALUES ARE";B;C;X$
```

Then type the word RUN. The computer will run the program again; but lines 15, 25, 35, and 45 make the computer print everything it's thinking. Check what the computer prints. If the computer prints what you expect in lines 15 and 25, but prints strange values in line 35 (or doesn't even get to line 35), you know the bug occurs before line 35 but after line 25; so the bug must be in line 30.

If your program contains hundreds of lines, you might not have the patience to insert lines 15, 25, 35, 45, 55, 65, 75, etc. Here's a short-cut....

Halfway down your program, insert a line that says to print all the values. Then run your program. If the line you inserted prints the correct values, you know the bug lies underneath that line; but if the line prints *wrong* values (or if the computer never reaches that line), you know the bug lies *above* that line. In either case, you know which half of your program contains the bug. In that half of the program, insert more lines, until you finally zero in on the line that contains the bug.

Shorten the Program

When all else fails, shorten the program.

Finding a bug in a program is like finding a needle in a haystack: the job is easier if the haystack is smaller. So make your program shorter: delete the last half of your program. Then run the shortened version. That way, you'll find out whether the first half of your program is working the way it's supposed to. When you've perfected the first half of your program, tack the second half back on.

Does your program contain a statement whose meaning you're not completely

sure of? Check the meaning, by reading a book or asking a friend; or write a tiny experimental program that contains the statement, and see what happens when you type RUN.

Hint: before you shorten your program (or write tiny experimental ones), SAVE the original version, even though it contains a bug. After you've played with the shorter versions, retrieve the original and fix it.

The easiest way to write a long, correct program is to write a short program first, debug it, then add a few more lines, debug them, add a few more lines, debug them, etc. In other words, start with a small program, perfect it, and then gradually add perfected extras, so you *gradually* build a perfected masterpiece. If you try to compose a long program all at once— instead of building it from perfected pieces— you'll have nothing more than a master*mess*— full of bugs.

Moral: to build a large masterpiece, start with a *small* masterpiece. Putting it another way: to build a skyscraper, begin by laying a good foundation; double-check the foundation, before you start adding the walls and the roof.

Decent Computers

From now on, I'll assume you're using a "decent" computer. Specifically, I'll assume:

If you have a Radio Shack computer, you're using Level-2 BASIC (*not* Level-1 BASIC).
If you have an Apple computer, you're using Applesoft BASIC (*not* Integer BASIC).
If you have a North Star computer, you're using Microsoft BASIC (*not* North Star's).
If you have a CP/M computer, you're using Microsoft BASIC (*not* CBASIC or BASIC-E).
If you have a PDP computer, you're using a PDP-10 or PDP-11 or PDP-20 (*not* a PDP-8).
If you have a PDP-11 or PDP-20 computer, you're using BASIC-PLUS or BASIC-PLUS-2.

Editing

To list your entire program, just type:

```
LIST
```

Here's how to list just line 20:

Microcomputers, PDP, Nova, Eclipse	Univac	HP-2000
LIST 20	LIST 20,20	LIST-20,20

Here's how to list from line 20 to line 50:

Microcomputers, PDP	Nova, Eclipse, Univac	HP-2000
LIST 20-50	LIST 20,50	LIST-20,50

Here's how to list from line 20 to the end:

Micros, PDP-10	PDP-11	PDP-20	Nova, Eclipse	Univac	HP-2000	CDC
LIST 20-	LIST 20-32767	LIST 20-99999	LIST 20,9999	LIST 20	LIST-20	LIST,20

Here's how to list from the beginning to line 20:

Microcomputers	PDP	Nova, Eclipse	Univac	HP-2000
LIST -20	LIST 1-20	LIST ,20	LIST 1,20	LIST-1,20

On PDP computers, you can list lines 20, 50, and 90, by saying:

```
LIST 20,50,90
```

To delete line 20, just type:

```
20
```

To delete lines 20-50, type this:

Radio Shack, PDP	Apple	Univac	HP-2000
DEL 20-50	DEL 20,50	EDIT DELETE 20,50	DEL-20,50

To delete (scratch out) *all* the lines in the program, type this:

Microcomputers, Nova, Eclipse, Univac	PDP-11	PDP-10, PDP-20, HP-2000, CDC
NEW	DEL	SCR

Suppose you've written a sloppy program, like this:

```
37 PRINT "SNARL"
68 PRINT "SMIRK"
91 GO TO 37
```

You can make the computer change the line numbers to 500, 510, and 520, so the program looks like this:

```
500 PRINT "SNARL"
510 PRINT "SMIRK"
520 GO TO 500
```

That's called *resequencing* or *renumbering*. Here's how to make the computer do it:

Radio Shack using BASICR	PDP-10, PDP-20	Nova, Eclipse	HP-2000	CDC
NAME 500	RES 500	RENUMBER 500	REN-500	RES,500

After you give that command, type the word LIST. Then the computer will show you the new version of your program. To save the new version on disk, type SAVE.

In that command, the "500" tells the computer to begin numbering at 500. If you omit the "500", here's where the computer begins:

Radio Shack using BASICR, PDP-10, Nova, Eclipse, HP-2000	PDP-20, CDC
It begins at 10.	It begins at 100.

Here's how to make line 68 become 500, and make the later lines become 510, 520, 530, 540, etc.:

Radio Shack using BASICR	PDP-10, PDP-20	HP-2000
NAME 500,68	RES 500,68	REN-500,10,68

Freezing

If your terminal uses a screen (instead of paper), the terminal might start flashing information on the screen so quickly that you can't read it— the information becomes just a blur that races past your eyes. (This happens especially when you tell the computer to list a long program, or when you run a program that says to PRINT in a loop.)

To make the computer pause, so you can read the information, you can give the computer an abortion. But abortions work slowly— by the time you finish giving the abortion, and the computer finishes reacting to it, some vital information might have raced past your eyes and off the screen. To make the computer stop printing *immediately*, freeze the screen, by doing this:

Computer	How to freeze the screen
Radio Shack	While you hold down the SHIFT key, tap the @ key.
Apple 2-plus, PDP	While you hold down the CONTROL key, tap the S key.

After you've frozen the screen, read what's on it. Then you have two choices: either give an abortion, or make the computer resume the printing. Here's how to make the computer resume the printing:

Computer	How to make it resume the printing
Radio Shack	While you hold down the SHIFT key, tap the @ key.
Apple 2-plus	While you hold down the CONTROL key, tap the S key.
PDP	While you hold down the CONTROL key, tap the Q key.

Paired Statements

DATA . . . READ

To make the computer handle a list, begin your program with the word DATA.

For example, suppose you want the computer to print this list of food:

```
MEAT
POTATOES
LETTUCE
TOMATOES
BUTTER
CHEESE
ONIONS
PEAS
```

Begin your program by saying:

```
10 DATA MEAT,POTATOES,LETTUCE,TOMATOES,BUTTER,CHEESE,ONIONS,PEAS
```

You must tell the computer to READ the DATA:

```
20 READ A$
```

Line 20 makes the computer read the first datum (which is MEAT), and call it A$. So A$ is MEAT. This line makes the computer print MEAT:

```
30 PRINT A$
```

This line makes the computer handle the rest of the data:

```
40 GO TO 20
```

Line 40 makes the computer go back to line 20, which reads the next datum (POTATOES). Altogether, the program looks like this:

```
10 DATA MEAT,POTATOES,LETTUCE,TOMATOES,BUTTER,CHEESE,ONIONS,PEAS
20 READ A$
30 PRINT A$
40 GO TO 20
```

Lines 20-40 form a *loop*. (Like most loops, the loop's bottom line says GO TO. Since the loop's top line says READ, the loop is called a *READ loop*.) The computer goes round and round the loop. Each time the computer comes to line 20, it reads another datum: the first time it comes to line 20, it reads MEAT; the next time, it reads POTATOES; the next time, it reads LETTUCE; the next time, it reads TOMATOES; etc. Line 30 makes the computer PRINT what it's read. Altogether, the computer will print:

```
MEAT
POTATOES
LETTUCE
TOMATOES
BUTTER
CHEESE
ONIONS
PEAS
```

After the computer prints PEAS, it comes to line 40 again, which makes it go back to line 20 again, so the computer tries to read more data again; but no more data remains! So the computer says:

```
OUT OF DATA
```

Then the computer stops.

So the last three lines the computer prints are:

```
ONIONS
PEAS
OUT OF DATA
```

Instead of saying OUT OF DATA, let's make the computer say THOSE ARE MY FAVORITE FOODS, so that the last three lines look like this:

```
ONIONS
PEAS
THOSE ARE MY FAVORITE FOODS
```

Here's how....At the end of the data, say END:

```
10 DATA MEAT,POTATOES,LETTUCE,TOMATOES,BUTTER,CHEESE,ONIONS,PEAS,END
```

When the computer reads the END, make the computer say THOSE ARE MY FAVORITE FOODS and then stop:

```
20 READ A$: IF A$="END" THEN PRINT "THOSE ARE MY FAVORITE FOODS": STOP
```

The END at the end of the data is called the *end mark*, because it marks the end. The routine that says—

```
IF A$="END" THEN PRINT "THOSE ARE MY FAVORITE FOODS": STOP
```

is called the *end routine*, because the computer does that routine at the end.

That program prints one copy of the computer's favorite foods. If you want the computer to print *many* copies, change line 20 to this:

```
20 READ A$: IF A$="END" THEN PRINT "THOSE ARE MY FAVORITE FOODS": RESTORE: GO TO 20
```

The word RESTORE tells the computer to go back to the beginning of the data. The computer will print:

```
MEAT
POTATOES
LETTUCE
TOMATOES
BUTTER
CHEESE
ONIONS
PEAS
THOSE ARE MY FAVORITE FOODS
MEAT
POTATOES
LETTUCE
TOMATOES
BUTTER
CHEESE
ONIONS
PEAS
THOSE ARE MY FAVORITE FOODS
MEAT
POTATOES
etc.
```

The computer will print copies of its list of foods again and again, forever, unless you give it an abortion.

Most practical computer programs involve a list of data. Your job, as a programmer, is to find out what the data is, and to begin your program by saying DATA. After typing the data, the next statement in your program should say READ. Farther down in your program, you should say GO TO, so that you create a READ loop. Your program consists of two parts: the DATA, and the READ loop.

Quotation Marks

On IBM, Nova, and Eclipse computers, you must put quotation marks around each data string, like this:

```
10 DATA "MEAT","POTATOES","LETTUCE","TOMATOES","BUTTER","CHEESE","ONIONS","PEAS","END"
```

Most other computers let you omit the quotation marks, unless the data string is unusual. Here are the details....

PDP-10: you must put quotation marks around a data string, if the string contains a comma or apostrophe or begins with a digit.

PDP-11: you must put quotation marks around a data string, if the string contains a comma or a blank space.

Most microcomputers: you must put quotation marks around a data string, if the string contains a comma or a colon.

When in doubt— or whenever a data statement doesn't seem to be working— insert quotation marks. They can't hurt.

Debts

Suppose these people owe you things:

Person	What the person owes
Bob	$537.29
Mike	a dime
Frank	2 golf balls
Harry	a steak dinner at Mario's
Mommy	a kiss

Let's remind those people of their debts, by writing them letters, in this form:

```
DEAR _____
     person's name

     I JUST WANT TO REMIND YOU...

THAT YOU STILL OWE ME ____
                      debt
```

Begin with the DATA:

```
10 DATA BOB,$537.29,MIKE,A DIME,FRANK,2 GOLF BALLS
20 DATA HARRY,A STEAK DINNER AT MARIO'S,MOMMY,A KISS
```

Since the data is too long to fit on a single line, I've put part of the data in line 10, and the rest in line 20. Each line of data must begin with the word DATA. In each line, put commas between the items. Do *not* put a comma at the end of the line. To be safe, you can insert quotation marks:

```
10 DATA "BOB","$537.29","MIKE","A DIME","FRANK","2 GOLF BALLS"
20 DATA "HARRY","A STEAK DINNER AT MARIO'S","MOMMY","A KISS"
```

(On most microcomputers, the quotation marks are unnecessary.)

The data comes in pairs: the first pair consists of BOB and $537.29. Tell the computer to READ each pair of DATA:

```
30 READ P$,D$
```

Line 30 makes the computer read the first pair of data; so P$ is the first person (BOB), and D$ is his debt ($537.29). These lines print the letter:

```
40 PRINT "DEAR ";P$
50 PRINT "     I JUST WANT TO REMIND YOU..."
60 PRINT "THAT YOU STILL OWE ME ";D$
```

At the end of the letter, leave two blank lines, to make room for your signature:

```
70 PRINT
80 PRINT
```

Then complete the READ loop, by making the computer go back to the beginning of the loop, to read the next pair:

```
90 GO TO 30
```

The computer will print a letter to each person, like this:

```
DEAR BOB,
     I JUST WANT TO REMIND YOU...
THAT YOU STILL OWE ME $537.29


DEAR MIKE,
     I JUST WANT TO REMIND YOU...
THAT YOU STILL OWE ME A DIME
```

etc.

After printing all the letters, the computer will say:

```
OUT OF DATA
```

Instead of saying OUT OF DATA, let's make the computer say:

```
I'VE FINISHED WRITING THE LETTERS
```

To do that, put END at the end of the data, *twice*—

```
20 DATA "HARRY","A STEAK DINNER AT MARIO'S","MOMMY","A KISS","END","END"
```

and say what to do when the computer reaches the ENDs:

```
30 READ P$,D$: IF P$="END" THEN PRINT "I'VE FINISHED WRITING THE LETTERS": STOP
```

You need *two* ENDs at the end of the data, because the READ statement says to read two strings (P$ and D$).

Diets

Suppose you're running a diet clinic, and get these results:

Person	Weight before	Weight after
Joe	273 pounds	219 pounds
Mary	412 pounds	371 pounds
Bill	241 pounds	173 pounds
Sam	309 pounds	198 pounds

Here's how to make the computer print a nice report....

Begin by feeding it the DATA:

```
10 DATA JOE,273,219,MARY,412,371,BILL,241,173,SAM,309,198
```

If your computer requires quotes around string data, rewrite the DATA like this:

```
10 DATA "JOE",273,219,"MARY",412,371,"BILL",241,173,"SAM",309,198
```

The data comes in triplets: the first triplet consists of JOE, 273, and 219. Tell the computer to READ each triplet of DATA:

```
20 READ N$,B,A
```

That line makes the computer read the first triplet of data; so N$ is the first person's name (JOE), B is his weight before (273), and A is his weight after (219). These lines print the report about him:

```
30 PRINT N$;" WEIGHED";B;"POUNDS BEFORE ATTENDING THE DIET CLINIC"
40 PRINT "BUT WEIGHED ONLY";A;"POUNDS AFTERWARDS"
50 PRINT "THAT'S A LOSS OF";B-A;"POUNDS"
```

At the end of the report about him, leave a blank line:

```
60 PRINT
```

Then complete the READ loop, by making the computer go back to the loop's beginning:

```
70 GO TO 20
```

The computer will print:

```
JOE WEIGHED 273 POUNDS BEFORE ATTENDING THE DIET CLINIC
BUT WEIGHED ONLY 219 POUNDS AFTERWARDS
THAT'S A LOSS OF 54 POUNDS

MARY WEIGHED 412 POUNDS BEFORE ATTENDING THE DIET CLINIC
BUT WEIGHED ONLY 371 POUNDS AFTERWARDS
THAT'S A LOSS OF 41 POUNDS
```

etc.

At the end, the computer will say OUT OF DATA. Instead, let's make it say:

```
COME TO OUR DIET CLINIC!
```

To do that, put an END and two zeros at the end of the data—

```
10 DATA "JOE",273,219,"MARY",412,371,"BILL",241,173,"SAM",309,198,"END",0,0
```

and say what to do when the computer reaches the END:

```
20 READ N$,B,A: IF N$="END" THEN PRINT "COME TO OUR DIET CLINIC!": STOP
```

You need the two zeros after the END, because the READ statement says to read two numbers (B and A) after the string N$. If you omit the zeros, the computer will say OUT OF DATA. If you hate zeros, you can use other numbers instead; but most programmers prefer zeros.

French Colors

Let's make the computer translate colors into French. For example, if the human says RED, we'll make the computer say the French equivalent, which is:

```
ROUGE
```

Altogether, a run will look like this:

```
RUN
WHICH COLOR INTERESTS YOU? RED
IN FRENCH, IT'S ROUGE
```

The program begins simply:

```
10 INPUT "WHICH COLOR INTERESTS YOU";C$
```

Next, we must make the computer translate the color into French. To do that, feed the computer this English-French dictionary:

English	French
white	blanc
yellow	jaune
orange	orange
red	rouge
green	vert
blue	bleu
brown	brun
black	noir

That dictionary becomes the data:

```
20 DATA WHITE,BLANC,YELLOW,JAUNE,ORANGE,ORANGE,RED,ROUGE
30 DATA GREEN,VERT,BLUE,BLEU,BROWN,BRUN,BLACK,NOIR
```

The data comes in pairs: the first pair consists of WHITE and BLANC. Tell the computer to READ each pair of DATA:

```
40 READ E$,F$
```

That line makes the computer read the first pair of data; so E$ is the first English color (WHITE), and F$ is its French equivalent (BLANC).

But that pair of data might be the *wrong pair*; for example, if the human requested RED, the human does *not* want the pair of WHITE and BLANC; instead, the human wants the pair of RED and ROUGE. So you must tell the computer: if the human's input doesn't match the English in the pair, go read another pair. Here's how to say that:

```
50 IF E$<>C$ THEN GO TO 40
```

That line says: if E$ (which is the English in the data) is not C$ (the color the human requested), go to 40 (which reads another pair).

Lines 40 and 50 form a loop; the computer goes round and round the loop, until it finds the pair of data that matches the human's request. Since the purpose of the loop is to search for data that matches, it's called a *search loop*.

After the computer has found the correct English-French pair, make the computer print the French:

```
60 PRINT "IN FRENCH, IT'S ";F$
```

Altogether, the program looks like this:

```
Ask the human→→→→→→→→→→→→→→10 INPUT "WHICH COLOR INTERESTS YOU";C$
Use this dictionary→→→→→→→→20 DATA WHITE,BLANC,YELLOW,JAUNE,ORANGE,ORANGE,RED,ROUGE
                           30 DATA GREEN,VERT,BLUE,BLEU,BROWN,BRUN,BLACK,NOIR
Look at the dictionary→→→→→40 READ E$,F$
If not found, look again→→→50 IF E$<>C$ THEN GO TO 40
Print the French→→→→→→→→→→→60 PRINT "IN FRENCH, IT'S ";F$
```

Here's a sample run:

```
RUN
WHICH COLOR INTERESTS YOU? RED
IN FRENCH, IT'S ROUGE
```

Here's another:

```
RUN
WHICH COLOR INTERESTS YOU? BROWN
IN FRENCH, IT'S BRUN
```

Here's another:

```
RUN
WHICH COLOR INTERESTS YOU? PINK
OUT OF DATA
```

The computer says OUT OF DATA because it can't find PINK in the data. Instead of saying OUT OF DATA, it would be nicer to say I WASN'T TAUGHT THAT COLOR.

We want the computer to say I WASN'T TAUGHT THAT COLOR, when it reaches the end of the data. To do that, say END at the end of the data; and when the computer reaches the END, tell it to say I WASN'T TAUGHT THAT COLOR:

```
            10 INPUT "WHICH COLOR INTERESTS YOU";C$
            20 DATA WHITE,BLANC,YELLOW,JAUNE,ORANGE,ORANGE,RED,ROUGE
Revised line→30 DATA GREEN,VERT,BLUE,BLEU,BROWN,BRUN,BLACK,NOIR,END,END
Revised line→40 READ E$,F$: IF E$="END" THEN PRINT "I WASN'T TAUGHT THAT COLOR": STOP
            50 IF E$<>C$ THEN GO TO 40
            60 PRINT "IN FRENCH, IT'S ";F$
```

After line 60, the program just ends. Instead of letting the computer end, let's make it automatically rerun the program and translate another color. To do that, say GO TO and RESTORE:

```
 10 INPUT "WHICH COLOR INTERESTS YOU";C$
 20 DATA WHITE,BLANC,YELLOW,JAUNE,ORANGE,ORANGE,RED,ROUGE
 30 DATA GREEN,VERT,BLUE,BLEU,BROWN,BRUN,BLACK,NOIR,END,END
→→40 READ E$,F$: IF E$="END" THEN PRINT "I WASN'T TAUGHT THAT COLOR": GO TO 70
 50 IF E$<>C$ THEN GO TO 40
 60 PRINT "IN FRENCH, IT'S ";F$
→→70 RESTORE
→→80 GO TO 10
```

FOR . . . NEXT

Let's make the computer print every number from 1 to 100, like this:

```
1
2
3
4
5
6
7
etc.
100
```

To do that, type this line—

```
20     PRINT X
```

and also tell the computer that you want X to be every number from 1 to 100. To tell the computer that, say "FOR X = 1 TO 100", like this:

Insert this line→→
```
10 FOR X = 1 TO 100
20     PRINT X
```

Whenever you write a program that contains the word FOR, you must say NEXT. So your program should look like this:

```
10 FOR X = 1 TO 100
20     PRINT X
```
Insert this line→→
```
30 NEXT X
```

That program works; it makes the computer print every number from 1 to 100.

Here's how it works....The computer begins at line 10, which says that you want X to be every number from 1 to 100. So X starts at 1. Then the computer comes to line 20, which says to print X; so the computer prints:

```
1
```

Then the computer comes to line 30, which says to do the same thing for the next X, and for the next X, and for the next X; so the computer prints 2, and 3, and 4, and so on, all the way up to 100.

That program makes the computer print many numbers. That's because the computer does line 20 many times (once for each X). The computer does line 20 many times because that line is between the words FOR and NEXT: it is underneath FOR, and above NEXT. The computer repeats anything that's between the words FOR and NEXT.

Most programmers get in the habit of indenting everything that comes between the words FOR and NEXT; so in that program, I indented the statement that says PRINT X. To make the indentation, you can hit the space bar repeatedly. Here's an easier way to indent:

Computer	How to indent
Radio Shack	Press the key that has a right-arrow (→).
PDP (and many others)	Press the TAB key— if your terminal has one. If it doesn't, hold down the CONTROL key, while you tap the I key.

Party

For a more advanced example, let's make the computer print these lyrics:

```
I SAW 2 MEN
MEET 2 WOMEN
TRA-LA-LA!

I SAW 3 MEN
MEET 3 WOMEN
TRA-LA-LA!

I SAW 4 MEN
MEET 4 WOMEN
TRA-LA-LA!

I SAW 5 MEN
MEET 5 WOMEN
TRA-LA-LA!

THEY ALL HAD A PARTY!
HA-HA-HA!
```

To do that, type these lines—

```
The first line of each verse→→→→→→→→→→20    PRINT "I SAW";X;"MEN"
The second line of each verse→→→→→→→→→30    PRINT "MEET";X;"WOMEN"
The third line of each verse→→→→→→→→→→40    PRINT "TRA-LA-LA!"
The blank line underneath each verse→→50    PRINT
```

and also tell the computer that you want X to be every number from 2 up to 5. Here's how:

```
Insert this line→→10 FOR X = 2 TO 5
                 20     PRINT "I SAW";X;"MEN"
                 30     PRINT "MEET";X;"WOMEN"
                 40     PRINT "TRA-LA-LA!"
                 50     PRINT
Insert this line→→60 NEXT X
```

At the end of the song, print the closing couplet:

```
                 10 FOR X = 2 TO 5
                 20     PRINT "I SAW";X;"MEN"
                 30     PRINT "MEET";X;"WOMEN"
                 40     PRINT "TRA-LA-LA!"
                 50     PRINT
                 60 NEXT X
Insert this line→→70 PRINT "THEY ALL HAD A PARTY!"
Insert this line→→80 PRINT "HA-HA-HA!"
```

That program works; it makes the computer print the entire song.

Here's an analysis:

The computer will do these lines repeatedly, for X=2, X=3, X=4, and X=5.

```
10 FOR X = 2 TO 5
20     PRINT "I SAW";X;"MEN"
30     PRINT "MEET";X;"WOMEN"
40     PRINT "TRA-LA-LA!"
50     PRINT
60 NEXT X
```

Then the computer will print this couplet, once.

```
70 PRINT "THEY ALL HAD A PARTY!"
80 PRINT "HA-HA-HA!"
```

Since the computer does lines 20-50 repeatedly, those lines form a loop. Here's the general rule: the statements between FOR and NEXT form a loop. The computer goes round and round the loop, for X=2, X=3, X=4, and X=5. Altogether, it goes around the loop 4 times, which is a finite number. Therefore, the loop is *finite*.

If you don't like the letter X, choose a different letter. For example, you can choose the letter I:

```
10 FOR I = 2 TO 5
20     PRINT "I SAW";I;"MEN"
30     PRINT "MEET";I;"WOMEN"
40     PRINT "TRA-LA-LA!"
50     PRINT
60 NEXT I
70 PRINT "THEY ALL HAD A PARTY!"
80 PRINT "HA-HA-HA!"
```

Most programmers prefer the letter I, when using the word FOR. In other words, most programmers say "FOR I", instead of "FOR X". Saying "FOR I" is an "old tradition". I'll follow the tradition: in the rest of this book, I'll say "FOR I", except in situations where some other letter feels peculiarly more natural.

Squares

To find the *square* of a number, multiply the number by itself. The square of 3 is "3 times 3", which is 9. The square of 4 is "4 times 4", which is 16.

Let's make the computer print the square of 3, 4, 5, etc., up to 100, like this:

```
THE SQUARE OF 3 IS 9
THE SQUARE OF 4 IS 16
THE SQUARE OF 5 IS 25
THE SQUARE OF 6 IS 36
THE SQUARE OF 7 IS 49
etc.
THE SQUARE OF 100 IS 10000
```

To do that, type this line—

```
20     PRINT "THE SQUARE OF";I;"IS";I*I
```

and also tell the computer that you want I to be every number from 3 up to 100. Here's the program:

```
10 FOR I = 3 TO 100
20     PRINT "THE SQUARE OF";I;"IS";I*I
30 NEXT I
```

Secret Meeting

This program prints 12 copies of the same message:

```
10 FOR I = 1 TO 12
20     PRINT "HUSH, HUSH!"
30     PRINT "  WE'RE HAVING A SECRET MEETING..."
40     PRINT "    IN THE COMPUTER ROOM..."
50     PRINT "      TONIGHT..."
```

```
60      PRINT "         AT 2 A.M."
70      PRINT "           WEAR A FUNNY HAT."
80      PRINT
90      PRINT
100 NEXT I
```

Lines 80 and 90 leave blank lines at the end of each copy, for your signature.

Kissing

This program kisses you as often as you like:

```
10 INPUT "HOW MANY TIMES SHOULD I KISS YOU";T
20 FOR I = 1 TO T
30     PRINT "I KISS YOU"
40 NEXT I
50 PRINT "NOW I KILL YOU"
```

Here's a sample run:

```
HOW MANY TIMES SHOULD I KISS YOU? 3
I KISS YOU
I KISS YOU
I KISS YOU
NOW I KILL YOU
```

Here's another run:

```
HOW MANY TIMES SHOULD I KISS YOU? 10
I KISS YOU
I KISS YOU
I KISS YOU
I KISS YOU
I KISS YOU
I KISS YOU
I KISS YOU
I KISS YOU
I KISS YOU
I KISS YOU
NOW I KILL YOU
```

What happens if you request *no* kisses? If you have a minicomputer or maxicomputer, the computer will obey you, and give you no kisses. But if you have a primitive microcomputer (such as the Radio Shack or Apple or PET), the computer will kiss you once, because those computers don't understand how to do anything less than once:

Most computers

```
HOW MANY TIMES SHOULD I KISS YOU? 0
NOW I KILL YOU
```

Primitive microcomputers

```
HOW MANY TIMES SHOULD I KISS YOU? 0
I KISS YOU
NOW I KILL YOU
```

Midnight

This program makes the computer count to midnight:

```
10 FOR I = 1 TO 12
20     PRINT I
30 NEXT I
40 PRINT "MIDNIGHT"
```

The computer will print:

```
 1
 2
```

```
 3
 4
 5
 6
 7
 8
 9
 10
 11
 12
MIDNIGHT
```

Let's put a semicolon at the end of line 20:

```
  10 FOR I = 1 TO 12
→→20     PRINT I;
  30 NEXT I
  40 PRINT "MIDNIGHT"
```

The semicolon makes the computer print each item on the same line, like this:

```
 1  2  3  4  5  6  7  8  9  10  11  12 MIDNIGHT
```

If you want the computer to press the RETURN key before MIDNIGHT, insert a PRINT line:

```
  10 FOR I = 1 TO 12
  20     PRINT I;
  30 NEXT I
→→35 PRINT
  40 PRINT "MIDNIGHT"
```

Line 35 makes the computer press the RETURN key just before MIDNIGHT, so the computer will print MIDNIGHT on a separate line, like this:

```
 1  2  3  4  5  6  7  8  9  10  11  12
MIDNIGHT
```

In line 20, the semicolon means: do *not* press the RETURN key after I. Line 35 means: *do* press the RETURN key. So line 35 undoes line 20, and makes the computer press the RETURN key before MIDNIGHT.

Let's make the computer count to midnight 3 times, like this:

```
 1  2  3  4  5  6  7  8  9  10  11  12
MIDNIGHT
 1  2  3  4  5  6  7  8  9  10  11  12
MIDNIGHT
 1  2  3  4  5  6  7  8  9  10  11  12
MIDNIGHT
```

To do that, put the entire program between the words FOR and NEXT:

```
→→5 FOR A = 1 TO 3
  10     FOR I = 1 TO 12
  20         PRINT I;
  30     NEXT I
  35     PRINT
  40     PRINT "MIDNIGHT"
→→50 NEXT A
```

That version contains a loop inside a loop: the loop that says "FOR I" is inside the loop that says "FOR A". The A loop is called the *outer loop*; the I loop is the *inner loop*. The inner loop's variable must differ from the outer loop's. Since we called the inner loop's variable "I", the outer loop's variable must *not* be called "I"; so I picked the letter A instead.

Often, programmers think of the outer loop as a bird's nest, and the inner loop as an egg *inside the nest*. So programmers say the inner loop is *nested in* the outer loop; the inner loop is a *nested loop*.

Favorite Color

This program plays a guessing game:

```
10 PRINT "I'LL GIVE YOU FIVE GUESSES...."
20 FOR I = 1 TO 5
30     INPUT "WHAT'S MY FAVORITE COLOR";G$
40     IF G$="PINK" THEN GO TO 100
50     PRINT "NO."
60 NEXT I
70 PRINT "SORRY, YOUR FIVE GUESSES ARE UP! YOU LOSE."
80 STOP

100 PRINT "CONGRATULATIONS! YOU DISCOVERED MY FAVORITE COLOR."
110 PRINT "IT TOOK YOU";I;"GUESSES."
```

Line 10 warns the human that only five guesses are allowed. Line 20 makes the computer count from 1 to 5; to begin, I is 1. Line 30 asks the human to guess the computer's favorite color; the guess is called G$. If the guess is PINK, the computer jumps from line 40 to line 100, prints CONGRATULATIONS, and tells how many guesses the human took. But if the guess is *not* PINK, the computer proceeds from line 40 to line 50, prints NO, and goes on to the next guess. If the human guesses five times without success, the computer proceeds from line 60 to line 70 and prints SORRY...YOU LOSE.

For example, if the human's third guess is PINK, the computer prints:

```
CONGRATULATIONS! YOU DISCOVERED MY FAVORITE COLOR.
IT TOOK YOU 3 GUESSES.
```

If the human's very first guess is PINK, the computer prints—

```
CONGRATULATIONS! YOU DISCOVERED MY FAVORITE COLOR.
IT TOOK YOU 1 GUESSES.
```

— which is ungrammatical but understandable.

Lines 20-60 form a loop. Line 20 says the loop will normally be done five times. The line after the loop, line 70, is the loop's *normal exit*. But if the human happens to input PINK, the computer jumps out of the loop early, to line 100, which is the loop's *abnormal exit*.

Short Cuts

On most microcomputers, when you say NEXT, you don't have to say the variable. So instead of saying—

```
50 NEXT I
```

you can say:

```
50 NEXT
```

Instead of saying—

```
10 FOR I = 1 TO 5
20     PRINT "FAT"
30     PRINT "CAT"
40     PRINT
50 NEXT I
```

you can put the entire loop on a single line, like this:

```
10 FOR I = 1 TO 5: PRINT "FAT": PRINT "CAT": PRINT: NEXT
```

Step

The FOR statement can be varied:

Statement	Meaning
`FOR I = 5 TO 17 STEP .1`	I will be every number from 5 to 17, counting by tenths. So I will be 5, then 5.1, then 5.2, then 5.3, etc., up to 17.
`FOR I = 5 TO 17 STEP 3`	I will be every third number from 5 to 17. So I will be 5, then 8, then 11, then 14, then 17.
`FOR I = 17 TO 5 STEP -3`	I will be every third number from 17 down to 5. So I will be 17, then 14, then 11, then 8, then 5.

To count down, you must use the word STEP. To count from 17 down to 5, give this instruction:

```
FOR I = 17 TO 5 STEP -1
```

This program prints a rocket countdown:

```
10 FOR I = 10 TO 1 STEP -1
20     PRINT I
30 NEXT I
40 PRINT "BLAST OFF!"
```

The computer will print:

```
 10
 9
 8
 7
 6
 5
 4
 3
 2
 1
BLAST OFF!
```

This statement is tricky:

```
FOR I = 5 TO 16 STEP 3
```

It says to start I at 5, and keep adding 3 until I gets past 16. So I will be 5, then 8, then 11, then 14. I won't be 17, since 17 is past 16. The first value of I is 5; the last value is 14.

In the statement FOR I = 5 TO 16 STEP 3, the *first value* or *initial value* of I is 5, the *limit value* is 16, and the *step size* or *increment* is 3. The I is called the *counter* or *index* or *loop-control variable*. Although the limit value is 16, the *last value* or *terminal value* is 14.

Programmers usually say "FOR I", instead of "FOR X", because the letter I reminds them of the word *index*.

Popular Features

Random Numbers

Usually, the computer is predictable: it does exactly what you say. But sometimes, you want the computer to be *un*predictable. For example, if you want to play a game of cards with the computer, and you want the computer to deal, you want the cards dealt to be unpredictable. If the cards were predictable— if you could figure out exactly which cards you'd be dealt, and which cards the computer would be dealt— the game would be boring.

In many other kinds of games too, you want the computer to be unpredictable, to "surprise" you. Without an element of surprise, the game would be boring.

And if you want the computer to act artistic, and create a new *original* masterpiece that's a "work of art", you need a way to make the computer get a "flash of inspiration". And flashes of inspiration are not predictable; they are surprises.

Here's how to make Radio Shack's computer act unpredictably. (I'll explain other computers later.)

This program makes the computer print an unpredictable number from 1 to 5:

```
10 RANDOM
20 PRINT RND(5)
```

Unpredictable numbers are called *random* numbers. Line 10 tells the computer that you want the numbers to be *completely* unpredictable, *completely* random. Line 20 makes the computer print a random number from 1 to 5. So the computer will print 1 or 2 or 3 or 4 or 5; you can't predict *which* of those numbers the computer will print; the computer's choice will be a surprise.

For example, when you type RUN, the computer might print 3. If you run the program a second time, the computer might print a different number (1 or 2 or 4 or 5), or it might print the same number (3). You can't predict which number the computer will print. The only thing you can be sure of is: the number will be from 1 to 5.

To make the computer print *many* such random numbers, say GO TO:

```
        10 RANDOM
        20 PRINT RND(5)
New line↔30 GO TO 20
```

The computer will print many numbers, like this:

```
3
2
4
4
1
3
5
2
2
5
etc.
```

Each number will be 1 or 2 or 3 or 4 or 5. The order in which the computer prints the numbers is unpredictable. The program is an infinite loop; to stop it, you must give an abortion. If you run the program again, the pattern will be different; for example, it might be:

```
1
4
3
3
2
5
1
1
2
etc.
```

That program consumes a lot of paper (or a lot of your screen). To use less paper (or less of your screen), make the computer print the numbers *across*, instead of down; make the computer print like this:

```
3  2  4  4  1  3  5  2  2  5  etc.
```

To do that, put a semicolon in the PRINT statement:

```
        10 RANDOM
New line→→20 PRINT RND(5);
        30 GO TO 20
```

That program prints random numbers up to 5. To see random numbers up to 1000, say RND(1000):

```
        10 RANDOM
New line→→20 PRINT RND(1000);
        30 GO TO 20
```

The computer will print something like this:

```
 485  729  8  537  1000  13  1  842  842  156  1000  972  etc.
```

This program plays a guessing game:

```
10 RANDOM
20 PRINT "I'M THINKING OF A NUMBER FROM 1 TO 100."
30 C=RND(100)
40 INPUT "WHAT DO YOU THINK MY NUMBER IS";G
50 IF G<C THEN PRINT "YOUR GUESS IS TOO LOW.": GO TO 40
60 IF G>C THEN PRINT "YOUR GUESS IS TOO HIGH.": GO TO 40
70 PRINT "CONGRATULATIONS!  YOU FOUND MY NUMBER!"
```

Line 20 makes the computer say:

```
I'M THINKING OF A NUMBER FROM 1 TO 100.
```

Line 30 makes the computer think of a random number from 1 to 100; the computer's number is called "C". Line 40 asks the human to guess the number; the guess is called "G". If the guess is less than the computer's number, line 50 makes the computer say "YOUR GUESS IS TOO LOW" and then GO TO 40, which lets the human guess again. If the guess is *greater* than the computer's number, line 60 makes the computer say "YOUR GUESS IS TOO HIGH" and then GO TO 40. When the human guesses correctly, the computer arrives at line 70, which prints:

```
CONGRATULATIONS!  YOU FOUND MY NUMBER!
```

Here's a sample run:

```
RUN
I'M THINKING OF A NUMBER FROM 1 TO 100.
WHAT DO YOU THINK MY NUMBER IS? 54
YOUR GUESS IS TOO LOW.
WHAT DO YOU THINK MY NUMBER IS? 73
YOUR GUESS IS TOO HIGH.
WHAT DO YOU THINK MY NUMBER IS? 62
YOUR GUESS IS TOO LOW.
WHAT DO YOU THINK MY NUMBER IS? 68
YOUR GUESS IS TOO LOW.
WHAT DO YOU THINK MY NUMBER IS? 70
YOUR GUESS IS TOO HIGH.
WHAT DO YOU THINK MY NUMBER IS? 69
CONGRATULATIONS!  YOU FOUND MY NUMBER!
```

This program makes the computer roll a pair of dice:

```
10 RANDOM
20 PRINT "I'M ROLLING A PAIR OF DICE"
30 A=RND(6)
40 PRINT "ONE OF THE DICE SAYS:";A
50 B=RND(6)
60 PRINT "THE OTHER SAYS:";B
70 PRINT "THE TOTAL IS";A+B
```

Line 20 makes the computer say:

```
I'M ROLLING A PAIR OF DICE
```

Each of the dice has 6 sides. Lines 30 and 40 roll one of the dice, by picking a number from 1 to 6. Lines 50 and 60 roll the other. Line 70 prints the total. Here's a sample run:

```
I'M ROLLING A PAIR OF DICE
ONE OF THE DICE SAYS: 3
THE OTHER SAYS: 5
THE TOTAL IS 8
```

Here's another run:

```
I'M ROLLING A PAIR OF DICE
ONE OF THE DICE SAYS: 6
THE OTHER SAYS: 4
THE TOTAL IS 10
```

This program makes the computer flip a coin:

```
10 RANDOM
20 R=RND(2)
30 IF R=1 THEN PRINT "HEADS": STOP
40 PRINT "TAILS"
```

Line 20 says R is a random number from 1 to 2; so R is either 1 or 2. If R is 1, line 30 makes the computer say HEADS; if R is 2, the computer skips line 30 and arrives at line 40, which prints TAILS instead. So the computer will print either HEADS or TAILS, depending on whether R is 1 or 2. Until you type RUN, you won't know which way the coin will flip; the choice is random. Each time you type RUN, the computer will flip the coin again; each time, the outcome is unpredictable.

Let's make the computer automatically re-run that program again and again, so that the computer automatically flips coins again and again, forever. To do that, put a loop in the program, by saying GO TO:

```
          10 RANDOM
          20 R=RND(2)
Revised line→30 IF R=1 THEN PRINT "HEADS ";: GO TO 20
Revised line→40 PRINT "TAILS ";: GO TO 20
```

The computer will print something like this:

```
HEADS TAILS HEADS HEADS TAILS TAILS HEADS TAILS TAILS TAILS HEADS TAILS HEADS etc.
```

This program makes the computer talk about your friends:

```
10 RANDOM
20 INPUT "TYPE THE NAME OF SOMEONE YOU LOVE...";N$
30 R=RND(3)
40 IF R=1 THEN PRINT N$;" HATES YOUR GUTS": GO TO 20
50 PRINT N$;" LOVES YOU, TOO": GO TO 20
```

Line 20 makes the computer ask:

```
TYPE THE NAME OF SOMEONE YOU LOVE...?
```

Suppose the human says SUZY. Then N$ is SUZY. Line 30 says R is a random number from 1 to 3. If R is 1, line 40 makes the computer say:

```
SUZY HATES YOUR GUTS
```

If R is 2 or 3, the computer arrives at line 50, which prints:

```
SUZY LOVES YOU, TOO
```

In that program, the chance is only 1 out of 3 that the computer will say

"HATES YOUR GUTS". The chance is 2 out of 3 that the computer will be nicer, and say "LOVES YOU, TOO". Here's a sample run:

```
RUN
TYPE THE NAME OF SOMEONE YOU LOVE...? SUZY
SUZY LOVES YOU, TOO
TYPE THE NAME OF SOMEONE YOU LOVE...? JOAN
JOAN HATES YOUR GUTS
TYPE THE NAME OF SOMEONE YOU LOVE...? ALICE
ALICE LOVES YOU, TOO
TYPE THE NAME OF SOMEONE YOU LOVE...? FRED
FRED LOVES YOU, TOO
TYPE THE NAME OF SOMEONE YOU LOVE...? UNCLE CHARLIE
UNCLE CHARLIE HATES YOUR GUTS
etc.
```

This program predicts what will happen to you today:

```
10 RANDOM
20 PRINT "YOU WILL HAVE A ";
30 R=RND(5)
40 IF R=1 THEN PRINT "WONDERFUL";
50 IF R=2 THEN PRINT "BETTER-THAN-AVERAGE";
60 IF R=3 THEN PRINT "SO-SO";
70 IF R=4 THEN PRINT "WORSE-THAN-AVERAGE";
80 IF R=5 THEN PRINT "TERRIBLE";
90 PRINT " DAY TODAY"
```

The computer will say—

```
YOU WILL HAVE A WONDERFUL DAY TODAY
```

or—

```
YOU WILL HAVE A TERRIBLE DAY TODAY
```

or some in-between comment. For inspiration, run that program when you get up in the morning.

All those programs work on Radio Shack's computer. If you have a different computer, you must make some changes. Instead of saying—

```
10 RANDOM
```

say this:

Computer	What to say
PDP	`10 RANDOMIZE\ DEF FND(N)=INT(1+N*RND)`
Nova, Eclipse	`10 RANDOMIZE: DEF FND(N)=INT(1+N*RND(0))`
IBM	`10 DEF FND(N)=INT(1+N*RND)`
HP-2000	`10 DEF FND(N)=INT(1+N*RND(0))`
Apple	`10 DEF FND(N)=INT(1+N*RND(1))`
PET, CBM	`10 R=RND(-TI): DEF FND(N)=INT(1+N*RND(1))`
Honeywell, Univac	`10 RANDOMIZE` `11 DEF FND(N)=INT(1+N*RND)`
CDC	`10 R=RND(CLK(0))` `11 DEF FND(N)=INT(1+N*RND(0))`

And on those computers, instead of saying RND, say FND, like this....

```
Radio Shack:  20 PRINT RND(5)
All others:   20 PRINT FND(5)
```

Zones

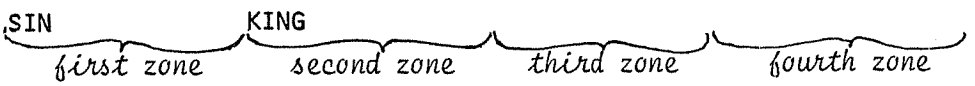
What are *zones*, and how do they work? The answer depends on what kind of computer you have. I'll begin by explaining Radio Shack's. Later, I'll explain how other computers differ. (The differences are minor.)

Radio Shack's terminal uses a screen instead of paper. The leftmost part of the screen is called the first zone; to the right of that zone lies the second zone; to the right of the second zone lies the third zone; and the rightmost part of the screen is called the fourth zone. So altogether, there are 4 zones. Each zone is 16 characters wide; the entire screen is 64 characters wide.

A comma makes the computer jump to a new zone; here's an example:

```
10 PRINT "SIN","KING"
```

The computer will print SIN and KING on the same line; but because of the comma before KING, the computer will print KING in the second zone, like this:



This program does the same thing:

```
10 PRINT "SIN",
20 PRINT "KING"
```

Line 10 makes the computer print SIN and then jump to the next zone. Line 20 makes the computer print KING. The computer will print:

```
SIN             KING
```

This program's silly:

```
10 PRINT "LOVE","CRIES","OUT"
```

The computer will print LOVE in the first zone, CRIES in the second zone, and OUT in the third zone, like this:

```
LOVE            CRIES           OUT
```

This program's even sillier:

```
10 PRINT "LOVE","CRIES","OUT","TO","ME","AT","NIGHT"
```

The computer will print LOVE in the first zone, CRIES in the second, OUT in the third, TO in the fourth, and the remaining words below, like this:

```
LOVE            CRIES           OUT             TO
ME              AT              NIGHT
```

This program makes the computer greet you:

```
10 PRINT "HI",
20 GO TO 10
```

The computer will print HI many times; each time will be in a new zone, like this:

```
HI              HI              HI              HI
HI              HI              HI              HI
HI              HI              HI              HI
HI              HI              HI              HI
HI              HI              HI              HI
etc.
```

This program prints a list of words and their opposites:

```
10 PRINT "GOOD","BAD"
20 PRINT "BLACK","WHITE"
30 PRINT "GRANDPARENT","GRANDCHILD"
40 PRINT "HE","SHE"
```

Line 10 makes the computer print GOOD, then jump to the next zone, then print BAD. Altogether, the computer will print:

```
GOOD            BAD
BLACK           WHITE
GRANDPARENT     GRANDCHILD
HE              SHE
```

The first zone contains a column of words; the second zone contains the opposites. Altogether, the computer's printing looks like a table. So whenever you want to make a table, use zones, by putting commas in your program.

Let's make the computer print this table:

```
NUMBER          SQUARE
 3               9
 4               16
 5               25
 6               36
 7               49
 8               64
 9               81
 10              100
```

Here's the program:

```
10 PRINT "NUMBER","SQUARE"
20 FOR I = 3 TO 10
30     PRINT I,I*I
40 NEXT I
```

Line 10 prints the word NUMBER at the top of the first column, and the word SQUARE at the top of the second. Line 20 says I goes from 3 to 10; to begin, I is 3. Line 30 makes the computer print:

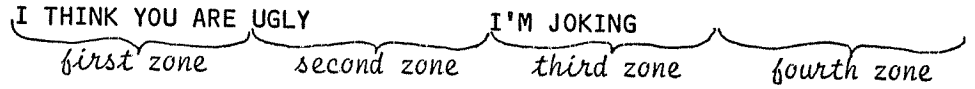
```
 3               9
```

Line 40 makes the computer do the same thing for the next I, and for the next I, and for the next; so the computer prints the whole table.

This program tells a bad joke:

```
10 PRINT "I THINK YOU ARE UGLY","I'M JOKING"
```

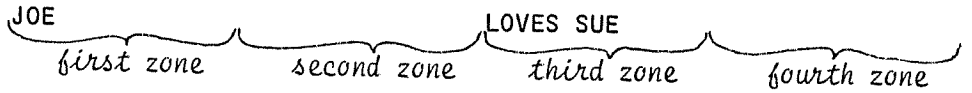
The computer will print I THINK YOU ARE UGLY, then jump to a new zone, then print I'M JOKING, like this:



You can make the computer skip over a zone:

```
10 PRINT "JOE"," ","LOVES SUE"
```

The computer will print JOE in the first zone, a blank space in the second zone, and LOVES SUE in the third zone, like this:



You can write that program even more briefly, like this:

```
10 PRINT "JOE",,"LOVES SUE"
```

When you combine commas with semicolons, you can get weird results:

```
10 PRINT "EAT","ME";"AT";"BALLS","NO";"W"
```

That line contains commas and semicolons. A comma makes the computer jump

to a new zone, but a semicolon does *not* make the computer jump. The computer will print EAT, then jump to a new zone, then print ME and AT and BALLS, then jump to a new zone, then print NO and W. Altogether, the computer will print:

```
EAT             MEATBALLS       NOW
```

Here's how other computers differ from Radio Shack's:

Computer	How many zones	Width of each zone	Width of entire screen or paper
Radio Shack	4 zones	16 characters	64 characters
PET, CBM	4 zones	10 characters	40 characters
Honeywell	5 zones	15 characters	75 characters
IBM	7 zones	18 characters	126 characters
Compucolor	8 zones	8 characters	64 characters
Apple	3 zones	usually 16 char. (rightmost zone has 8)	40 characters
PDP-10	5 zones	usually 14 char. (rightmost zone has 16)	72 characters
PDP-11, PDP-20	5 zones	usually 14 char. (rightmost zone has 24)	80 characters
Univac	5 zones	usually 15 char. (rightmost zone has 12)	72 characters
Nova, Eclipse	6 zones	usually 14 char. (rightmost zone has 10)	80 characters

Lesson 4

Tackle the Tough Stuff

Loop Techniques

Incrementing

Here's a strange program:

```
10 A=5
20 A=3+A
30 PRINT A
```

Line 20 means: the new A is 3 plus the old A. So the new A is 3+5, which is 8. Line 30 prints:

```
 8
```

Here's another weirdo:

```
10 B=6
20 B=B+1
30 PRINT B*2
```

Line 20 says the new B is "the old B plus 1". So the new B is 6+1, which is 7. Line 30 prints:

```
 14
```

In that program, line 10 says B is 6; but line 20 increases B, by adding 1 to B; so B becomes 7. Programmers say that B has been *increased* or *incremented*. In line 20, the "1" is called the *increase* or the *increment*.

Decrementing

The opposite of *increment* is *decrement*:

```
10 J=500
20 J=J-1
30 PRINT J
```

Line 10 says J starts at 500; but line 20 says the new J is "the old J minus 1"; so the new J is 500-1, which is 499. Line 30 prints:

```
 499
```

In that program, J was *decreased* (or *decremented*). In line 20, the "1" is called the *decrease* (or *decrement*).

Counting

Suppose you want the computer to count, starting at 3, like this:

```
3
4
5
6
7
8
```
etc.

This program does it, by a special technique:

```
10 C=3
20 PRINT C
30 C=C+1
40 GO TO 20
```

In that program, C is called the *counter*, because it helps the computer count. Line 10 says C starts at 3. Line 20 makes the computer print C; so the computer prints:

```
 3
```

Line 30 increases C, by adding 1 to it; so C becomes 4. Line 40 sends the computer back to line 20, which prints the new value of C:

```
 4
```

Then the computer comes to line 30 again, which increases C again; so C becomes 5. Line 40 sends the computer back to line 20 again, which prints:

```
 5
```

The program is an infinite loop: the computer will print 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, and so on, forever, unless you give an abortion.

Here's the general procedure for making the computer count:

1. Start C at some value (such as 3).
2. Use C. (For example, tell the computer to PRINT C.)
3. Increase C (by saying C=C+1).
4. GO back TO step 2.

To read the printing more easily, put a semicolon at the end of the PRINT statement:

```
  10 C=3
→→20 PRINT C;
  30 C=C+1
  40 GO TO 20
```

The semicolon makes the computer print horizontally:

```
 3  4  5  6  7  8  etc.
```

This program makes the computer count, starting at 1:

```
→→10 C=1
  20 PRINT C;
  30 C=C+1
  40 GO TO 20
```

The computer will print 1, 2, 3, 4, etc.

This program makes the computer count, starting at 0:

```
→→10 C=0
  20 PRINT C;
  30 C=C+1
  40 GO TO 20
```

The computer will print 0, 1, 2, 3, 4, etc.

Let's make the computer print a table, showing each number and its square, like this:

```
NUMBER          SQUARE
 0               0
 1               1
 2               4
```

```
3               9
4               16
5               25
6               36
7               49
8               64
etc.
```

This program does it:

```
→→5 PRINT "NUMBER","SQUARE"
  10 C=0
→→20 PRINT C,C*C
  30 C=C+1
  40 GO TO 20
```

Line 5 prints the headings. Lines 10-40 make the computer count (0, 1, 2, 3, etc.); but instead of printing just C, line 20 also prints C*C.

Let's make the computer print a table that shows the squares of decimals, like this:

```
NUMBER          SQUARE
 0               0
 .1              .01
 .2              .04
 .3              .09
 .4              .16
 .5              .25
 .6              .36
 .7              .49
 .8              .64
 .9              .81
 1               1
 1.1             1.21
 1.2             1.44
 1.3             1.69
 1.4             1.96
 1.5             2.25
 etc.
```

To make the computer do that, just change line 30, so that the computer increases by .1 instead of by 1:

```
  5 PRINT "NUMBER","SQUARE"
  10 C=0
  20 PRINT C,C*C
→→30 C=C+.1
  40 GO TO 20
```

Quiz

Let's make the computer give this quiz:

What's the capital of Nevada?
What's the chemical symbol for iron?
What word means 'brother or sister'?
What's Beethoven's first name?
How many cups are in a quart?

To make the computer score the quiz, we'll have to tell it the correct answers, which are:

Carson City
Fe
sibling

Ludwig
4

So our program will contain this data:

```
10 DATA "WHAT'S THE CAPITAL OF NEVADA","CARSON CITY"
20 DATA "WHAT'S THE CHEMICAL SYMBOL FOR IRON","FE"
30 DATA "WHAT WORD MEANS 'BROTHER OR SISTER'","SIBLING"
40 DATA "WHAT WAS BEETHOVEN'S FIRST NAME","LUDWIG"
50 DATA "HOW MANY CUPS ARE IN A QUART","4"
```

Tell the computer to READ the data:

```
100 READ Q$,A$
```

That line reads a pair of data: it reads a question (Q$) and the correct answer (A$). The next step is to make the computer ask the question, and wait for the human's response:

```
110 PRINT Q$;
120 INPUT "??";H$
```

Line 110 prints the question. Line 120 prints question marks after the question, and waits for the human to respond; the human's response is called H$.

To complete the program, evaluate the human's response. If the human's response (H$) is the correct answer (A$), make the computer say "CORRECT", and then GO TO the next question:

```
130 IF H$=A$ THEN PRINT "CORRECT": GO TO 100
```

But if the human's response is wrong, make the computer say "NO" and reveal the correct answer:

```
140 PRINT "NO, THE ANSWER IS: ";A$: GO TO 100
```

Here's a sample run:

```
RUN
WHAT'S THE CAPITAL OF NEVADA??? LAS VEGAS
NO, THE ANSWER IS: CARSON CITY
WHAT'S THE CHEMICAL SYMBOL FOR IRON??? FE
CORRECT
WHAT WORD MEANS 'BROTHER OR SISTER'??? I GIVE UP
NO, THE ANSWER IS: SIBLING
WHAT WAS BEETHOVEN'S FIRST NAME??? LUDVIG
NO, THE ANSWER IS: LUDWIG
HOW MANY CUPS ARE IN A QUART??? 4
CORRECT
OUT OF DATA
```

To give a quiz about different topics, change the data in lines 10-50.

Instead of making the computer say OUT OF DATA, let's make it say:

```
I HOPE YOU ENJOYED THE QUIZ
```

To do that, write an end mark and an end routine:

```
60 DATA "END","END"
100 READ Q$,A$: IF Q$="END" THEN PRINT "I HOPE YOU ENJOYED THE QUIZ": STOP
```

Let's make the computer count how many questions the human answered correctly. To do that, we need a counter. As usual, we'll call it C:

```
→→5 C=0
  10 DATA "WHAT'S THE CAPITAL OF NEVADA","CARSON CITY"
  20 DATA "WHAT'S THE CHEMICAL SYMBOL FOR IRON","FE"
  30 DATA "WHAT WORD MEANS 'BROTHER OR SISTER'","SIBLING"
```

```
  40 DATA "WHAT WAS BEETHOVEN'S FIRST NAME","LUDWIG"
  50 DATA "HOW MANY CUPS ARE IN A QUART","4"
  60 DATA "END","END"
→→100 READ Q$,A$: IF Q$="END" THEN PRINT "YOU ANSWERED";C;"OF THE QUESTIONS CORRECT
  LY": PRINT "I HOPE YOU ENJOYED THE QUIZ": STOP
  110 PRINT Q$;
  120 INPUT "??";H$
→→130 IF H$=A$ THEN PRINT "CORRECT": C=C+1: GO TO 100
  140 PRINT "NO, THE ANSWER IS: ";A$: GO TO 100
```

Line 5 begins the counter at 0 (because at the beginning, the human hasn't answered any questions correctly yet). When the program ends, line 100 prints the value of the counter. Line 130 makes sure that each time the human answers a question correctly, the counter increases.

Line 100 prints a message such as:

```
YOU ANSWERED 2 OF THE QUESTIONS CORRECTLY
```

It would be nicer to print—

```
YOU ANSWERED 2 OF THE 5 QUESTIONS CORRECTLY
YOUR SCORE IS 40 %
```

— or, if the quiz were changed to include 8 questions:

```
YOU ANSWERED 2 OF THE 8 QUESTIONS CORRECTLY
YOUR SCORE IS 25 %
```

To make the computer print the 5 or the 8, we must make the computer count how many questions were asked. So we need another counter. Since we already used C to count the number of correct answers, we'll use D to count the number of questions asked. Like C, D must start at 0; and we must increase D, by adding 1 each time another question is asked:

```
→→5 C=0: D=0
  10 DATA "WHAT'S THE CAPITAL OF NEVADA","CARSON CITY"
  20 DATA "WHAT'S THE CHEMICAL SYMBOL FOR IRON","FE"
  30 DATA "WHAT WORD MEANS 'BROTHER OR SISTER'","SIBLING"
  40 DATA "WHAT WAS BEETHOVEN'S FIRST NAME","LUDWIG"
  50 DATA "HOW MANY CUPS ARE IN A QUART","4"
  60 DATA "END","END"
→→100 READ Q$,A$: IF Q$="END" THEN PRINT "YOU ANSWERED";C;"OF THE";D;"QUESTIONS COR
  RECTLY": PRINT "YOUR SCORE IS";C/D*100;"%": PRINT "I HOPE YOU ENJOYED THE QUIZ":
  STOP
  110 PRINT Q$;
→→115 D=D+1
  120 INPUT "??";H$
  130 IF H$=A$ THEN PRINT "CORRECT": C=C+1: GO TO 100
  140 PRINT "NO, THE ANSWER IS: ";A$: GO TO 100
```

Summing

Let's make the computer imitate an adding machine, so a run looks like this:

```
RUN
NOW THE SUM IS 0
WHAT NUMBER DO YOU WANT TO ADD TO THE SUM? 5
NOW THE SUM IS 5
WHAT NUMBER DO YOU WANT TO ADD TO THE SUM? 3
NOW THE SUM IS 8
WHAT NUMBER DO YOU WANT TO ADD TO THE SUM? 6.1
NOW THE SUM IS 14.1
WHAT NUMBER DO YOU WANT TO ADD TO THE SUM -10
```

```
NOW THE SUM IS 4.1
```
etc.

Here's the program:

```
10 S=0
20 PRINT "NOW THE SUM IS";S
30 INPUT "WHAT NUMBER DO YOU WANT TO ADD TO THE SUM";X
40 S=S+X
50 GO TO 20
```

Line 10 starts the sum at 0. Line 20 prints the sum. Line 30 asks the human what number to add to the sum; the human's number is called X. Line 40 adds X to the sum, so the sum changes. Line 50 makes the computer go to line 20, which prints the new sum. Lines 20-50 form an infinite loop, which you must abort.

Here's the general procedure for making the computer find a sum:

1. Start S at 0.
2. Use S. (For example, tell the computer to PRINT S.)
3. Find out what number to add to S. (For example, tell the human to input a number X.)
4. Increase S (by saying S = S + the number to be added).
5. GO back TO step 2.

Checking Account

If your bank is nasty, it treats your checking account like this: you must pay a 10¢ service charge for each good check; you must pay a $5 penalty for each check that bounces; and the money you've deposited earns no interest.

This program makes the computer imitate such a bank:

Start the sum at 0→→→
```
10 S=0
```
Chat with the human→→
```
20 PRINT "YOUR CHECKING ACCOUNT CONTAINS";S
30 INPUT "DEPOSIT OR WITHDRAW";A$
40 IF A$="DEPOSIT" THEN GO TO 100
50 IF A$="WITHDRAW" THEN GO TO 200
60 PRINT "PLEASE SAY 'DEPOSIT' OR 'WITHDRAW'": GO TO 30
```
Deposit some money→→→
```
100 INPUT "HOW MUCH DO YOU WANT TO DEPOSIT";D
110 S=S+D
120 GO TO 20
```
Withdraw some money→→
```
200 INPUT "HOW MUCH DO YOU WANT TO WITHDRAW";W
210 W=W+.10
220 IF W<=S THEN PRINT "OKAY": S=S-W: GO TO 20
230 PRINT "THAT CHECK BOUNCED": S=S-5: GO TO 20
```

Line 10 starts the sum at 0. Line 20 prints the sum. Line 30 asks whether the human wants to deposit or withdraw.

If the human say DEPOSIT, the computer goes from line 40 to line 100, which asks how much to deposit. Line 110 increases the sum in the account, by adding the deposit. Line 120 sends the computer back to line 20, whcih tells the human the new sum (i.e., the balance), and gets ready for the next transaction.

If the human says WITHDRAW instead of deposit, the computer goes from line 50 to line 200, which asks how much to withdraw. Line 210 adds the 10¢ charge to the withdrawal amount. Line 220 compares the withdrawal amount (W) against the sum in the account (S). If W<=S, the computer says OKAY, processes the check, decreases the sum in the account (by subtracting W), and gets ready for the next transaction (by going back to line 20). If W>S instead, the computer says THAT CHECK BOUNCED, decreases the sum in the account by $5 (the penalty fee), and goes back to line 20, for the next transaction.

That program will annoy some customers. For example, suppose you have $1

in your account, and you try to write a check for 95¢. Since 95¢ + the 10¢ service charge = $1.05, which is more than you have in your account, your check will bounce, and you'll be penalized $5, so your balance will become *negative* $4, and the bank will demand that *you* pay the *bank* $4— just because you wrote a check for 95¢!

Another nuisance is when you leave town permanently, and want to close your account. If your account contains $1, you cannot get the dollar back! The most you can withdraw is 90¢, because 90¢ + the 10¢ service charge = $1.

In those cases, the program is "nasty", and makes the customers hate the bank— and hate the computer! The bank should change the program, to make it act friendlier, by inserting this line—

```
15 IF S<0 THEN S=0
```

and eliminating line 210 and retyping lines 220-230, as follows:

```
220 IF W<=S THEN PRINT "OKAY": S=S-W-.10: GO TO 15
230 PRINT "THAT CHECK BOUNCED": S=S-5: GO TO 15
```

Line 15 is kind, because it makes the computer change a negative sum to 0, so that the bank never accuses the customer of owing money. Eliminating line 110 is kind, because now the computer ignores the 10¢ service charge when comparing W against S in line 220. The new versions of lines 220 and 230 make sure that the computer goes to line 15 (the kind line) before arriving at line 20.

But some banks would argue that the new program is *too* kind! For example, if a person has only 1¢ in his account, and writes a check for a million dollars, which bounces, the new program charges him only 1¢ for the bad check; $5 might be more reasonable.

Moral: the hardest thing about programming is choosing your goal— deciding what you *want* the computer to do.

Betting

This program lets you bet:

```
10 RANDOM
20 S=100
30 PRINT "YOU HAVE";S;"DOLLARS."
40 INPUT "HOW MANY DOLLARS DO YOU WANT TO BET";B
50 IF B>S THEN PRINT "YOU DON'T HAVE THAT MUCH MONEY!  YOU MUST BET LESS.": GO TO 40
60 IF B<0 THEN PRINT "YOU CAN'T BET LESS THAN NOTHING!": GO TO 40
70 IF B=0 THEN PRINT "I GUESS YOU DON'T WANT TO BET ANYMORE.": GO TO 300
80 INPUT "DO YOU WANT TO BET ON HEADS OR TAILS";A$
90 IF A$="HEADS" THEN GO TO 200
100 IF A$="TAILS" THEN GO TO 200
110 PRINT "PLEASE SAY 'HEADS' OR 'TAILS'": GO TO 80
200 R=RND(2)
210 IF R=1 THEN R$="HEADS"
220 IF R=2 THEN R$="TAILS"
230 PRINT "THE COIN SAYS ";R$;"."
240 IF R$=A$ THEN PRINT "YOU WIN";B;"DOLLARS": S=S+B: GO TO 30
250 PRINT "YOU LOSE";B;"DOLLARS": S=S-B: IF S>0 THEN GO TO 30
260 PRINT "YOU'RE BROKE!  TOO BAD!"
300 PRINT "THANK YOU FOR PLAYING WITH ME.  YOU WERE FUN TO PLAY WITH."
310 PRINT "I HOPE YOU PLAY AGAIN SOMETIME."
```

(That program works on Radio Shack's computer. On other computers, change line 10; and in line 200, change RND to FND.)

Here's how the program works. Lines 20 and 30 make the computer say:

```
YOU HAVE 100 DOLLARS.
```

Line 40 makes the computer ask:

```
HOW MANY DOLLARS DO YOU WANT TO BET?
```

Lines 50-70 make sure the bet is reasonable. Line 80 makes the computer ask:

```
DO YOU WANT TO BET ON HEADS OR TAILS?
```

Lines 90-110 make sure the human says HEADS or TAILS.

Lines 200-230 make the computer flip a coin. Lines 240-250 determine whether the human won or lost the bet, and then send the computer back to line 30, for another round (if the human isn't broke yet). Lines 260-310 say good-bye to the human.

Here's a sample run:

```
YOU HAVE 100 DOLLARS.
HOW MANY DOLLARS DO YOU WANT TO BET? 120
YOU DON'T HAVE THAT MUCH MONEY!  YOU MUST BET LESS.
HOW MANY DOLLARS DO YOU WANT TO BET? 75
DO YOU WANT TO BET ON HEADS OR TAILS? HEADS
THE COIN SAYS TAILS.
YOU LOSE 75 DOLLARS.
YOU HAVE 25 DOLLARS.
HOW MANY DOLLARS DO YOU WANT TO BET? 10
DO YOU WANT TO BET ON HEADS OR TAILS? TAILS
THE COIN SAYS TAILS.
YOU WIN 10 DOLLARS.
YOU HAVE 35 DOLLARS.
HOW MANY DOLLARS DO YOU WANT TO BET? 35
DO YOU WANT TO BET ON HEADS OR TAILS? TAILS
THE COIN SAYS HEADS.
YOU LOSE 35 DOLLARS.
YOU'RE BROKE!  TOO BAD!
THANK YOU FOR PLAYING WITH ME.  YOU WERE FUN TO PLAY WITH.
I HOPE YOU PLAY AGAIN SOMETIME.
```

To make the output prettier, insert these lines:

```
30 PRINT
31 PRINT "YOU HAVE";S;"DOLLARS.  HERE THEY ARE:"
32 FOR I = 1 TO S
33     PRINT "$";
34 NEXT I
35 PRINT
```

Now the run looks like this:

```
RUN

YOU HAVE 100 DOLLARS.  HERE THEY ARE:
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
$$$$$$$$$$$$$$$$$$$$
HOW MANY DOLLARS DO YOU WANT TO BET? 120
YOU DON'T HAVE THAT MUCH MONEY!  YOU MUST BET LESS.
HOW MANY DOLLARS DO YOU WANT TO BET? 75
DO YOU WANT TO BET ON HEADS OR TAILS? HEADS
THE COIN SAYS TAILS.
YOU LOSE 75 DOLLARS.

YOU HAVE 25 DOLLARS.  HERE THEY ARE:
$$$$$$$$$$$$$$$$$$$$$$$$$
```

```
HOW MANY DOLLARS DO YOU WANT TO BET? 10
DO YOU WANT TO BET ON HEADS OR TAILS? TAILS
THE COIN SAYS TAILS.
YOU WIN 10 DOLLARS.

YOU HAVE 35 DOLLARS.  HERE THEY ARE:
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
HOW MANY DOLLARS DO YOU WANT TO BET? 35
DO YOU WANT TO BET ON HEADS OR TAILS? TAILS
THE COIN SAYS HEADS.
YOU LOSE 35 DOLLARS.
YOU'RE BROKE!  TOO BAD!
THANK YOU FOR PLAYING WITH ME.  YOU WERE FUN TO PLAY WITH.
I HOPE YOU PLAY AGAIN SOMETIME.
```

Series

Let's make the computer add together all the numbers from 7 to 100. In other words, let's make the computer find the sum of this series: 7+8+9+...+100. Here's how:

```
Start the Sum at 0→→→→→→→→→→→→→→→→→→→→→→→→10 S=0
Make I go from 7 to 100→→→→→→→→→→→→→→→→→→→20 FOR I = 7 TO 100
Increase the Sum, by adding each I to it→→30     S=S+I
                                          40 NEXT I
Print the final Sum (which is 5029)→→→→→→→50 PRINT S
```

Let's make the computer add together the *squares* of all the numbers from 7 to 100. In other words, let's make the computer find the sum of this series: (7 squared) + (8 squared) + (9 squared) + ... + (100 squared). Here's how:

```
          10 S=0
          20 FOR I = 7 TO 100
New line→→30     S=S+I*I
          40 NEXT I
          50 PRINT S
```

It's the same as the previous program, except that line 30 says to add I*I instead of I. Line 50 prints the final sum, which is 338259.

Data

This program adds together the numbers in the data:

```
10 S=0
20 DATA 5, 3, 6.1, etc.
30 DATA etc.
40 DATA etc.
50 DATA 0
60 READ X: IF X=0 THEN PRINT S: STOP
70 S=S+X
80 GO TO 60
```

Line 10 starts the sum at zero. Lines 20-40 contain the numbers to be added. The zero in line 50 is an end mark. Line 60 reads an X from the data. If the X it reads is zero, the end of the data has been reached, so we want the computer to print the sum (S) and stop; but if the X it reads is *not* zero, the computer proceeds from line 60 to line 70, adds X to the sum, and goes from line 80 to line 60, which reads another X.

Subscripts

Single Subscripts

Instead of being a single number, X can be a whole *list* of numbers, like this:

$$X=\begin{pmatrix} 57.2 \\ -8.3 \\ 476 \\ 2.008 \\ -19 \\ 372.402 \\ 0 \\ 215.6 \end{pmatrix}$$

Here's how to make X be that list of numbers....

Begin your program by saying:

```
10 DIM X(8)
```

That says X will be a list of 8 numbers. DIM means *dimension*; the line says the dimension of X is 8.

Next, tell the computer what numbers are in X. Type these lines:

```
20 X(1)=57.2
30 X(2)=-8.3
40 X(3)=476
50 X(4)=2.008
60 X(5)=-19
70 X(6)=372.402
80 X(7)=0
90 X(8)=215.6
```

Line 20 says: X's first number is 57.2. Line 30 says X's second value is -8.3. The remaining lines define the other numbers in X.

If you'd like the computer to print all those numbers, type this:

```
100 MAT PRINT X
```

That means: print all the numbers in X. The computer will print:

```
 57.2
-8.3
 476
 2.008
-19
 372.402
 0
 215.6
```

Unfortunately, most microcomputers don't understand the word MAT. So if you have a microcomputer, do *not* say:

```
100 MAT PRINT X
```

Instead, say this:

```
100 FOR I = 1 TO 8: PRINT X(I): NEXT
```

That makes the computer print X(1), and X(2), and X(3), etc. So the computer is doing a MAT PRINT.

In that program, line 20 talks about X(1). Instead of saying X(1), math books say:

X_1

The "1" is called a *subscript*. Similarly, in line 30, which says "X(2)=-8.3", the number 2 is a subscript. Some programmers pronounce line 30 like this: "X subscripted by 2 is -8.3". To be briefer, most programmers say this instead: "X sub 2 is -8.3". Some programmers simply say: "X 2 is -8.3".

In that program, X is called an *array* (or *matrix*). Definition: an *array* (or *matrix*) is a variable that has subscripts.

Line 100 says "MAT PRINT X". It means: the MATrix to be PRINTed is X.

Data

That program said X(1) is 57.2, and X(2) is -8.3, and so on. This program does the same thing, more briefly:

```
10 DIM X(8)
20 DATA 57.2, -8.3, 476, 2.008, -19, 372.402, 0, 215.6
30 MAT READ X
40 MAT PRINT X
```

Line 10 says X will be a list of 8 numbers. Line 20 contains a list of 8 numbers. Line 30 makes the computer READ those numbers, and call them X. Line 40 makes the computer print them.

In that program, line 30 says: the matrix to be read is X.

If you have a microcomputer that doesn't understand MAT, you cannot say—

```
30 MAT READ X
40 MAT PRINT X
```

Instead, say this:

```
30 FOR I = 1 TO 8: READ X(I): NEXT
40 FOR I = 1 TO 8: PRINT X(I): NEXT
```

In that program, the first three lines say:

DIM
DATA
MAT READ

Most practical programs begin with those three lines.

Let's make the program fancier, by adding extra lines. This line makes the computer add the 8 numbers together and print the sum:

```
50 PRINT X(1)+X(2)+X(3)+X(4)+X(5)+X(6)+X(7)+X(8)
```

That line makes the computer print:

```
 1095.91
```

Let's make the computer print the numbers in reverse order (starting with the 8th number, and ending with the first). In other words, let's make the computer print X(8), then print X(7), then print X(6), etc. To do that, you could say:

```
60 PRINT X(8)
70 PRINT X(7)
80 PRINT X(6)
```

etc.

But this way is shorter:

```
60 FOR I = 8 TO 1 STEP -1
70     PRINT X(I)
80 NEXT I
```

Lines 60-80 print:

```
 215.6
 0
 372.402
-19
 2.008
 476
-8.3
 57.2
```

Strange Example

Getting tired of X? Then pick another letter! For example, you can play with Y:

Silly, useless program	What the program means
`10 DIM Y(5)`	Y will be a list of 5 numbers.
`20 FOR I = 2 TO 5`	
`30  Y(I)=I*100`	Y(2) is 200. Y(3) is 300. Y(4) is 400. Y(5) is 500.
`40 NEXT I`	
`50 Y(1)=Y(2)-3`	Y(1) is 200-3; so Y(1) is 197.
`60 Y(3)=Y(1)-2`	Y(3) changes. It becomes 197-2, which is 195.
`70 MAT PRINT Y`	Print Y(1), Y(2), Y(3), Y(4), and Y(5).

If your computer doesn't understand MAT, replace line 70 by this:

```
70 FOR I = 1 TO 5: PRINT Y(I): NEXT
```

Line 70 prints:

```
 197
 200
 195
 400
 500
```

Analysis

Suppose you want to analyze 50 numbers. Begin your program by saying:

```
10 DIM X(50)
```

Then type the 50 numbers, as data, like this:

```
20 DATA etc.
30 DATA etc.
40 DATA etc.
```

Tell the computer to READ the data:

```
100 MAT READ X
```

If your computer doesn't understand MAT, say this instead:

```
100 FOR I = 1 TO 50: PRINT X(I): NEXT
```

After line 100, you have many choices, depending on which problem you want to solve....

Problem: print all the values of X. Solution:

```
110 FOR I = 1 TO 50
120    PRINT X(I)
130 NEXT I
```

Problem: print all the values of X, in reverse order. Solution:

```
110 FOR I = 50 TO 1 STEP -1
120    PRINT X(I)
130 NEXT I
```

Problem: print the sum of all the values of X. In other words, print X(1)+X(2)+X(3)+...+X(50). Solution: start the sum at 0—

```
110 S=0
```

and then increase the sum, by adding each X(I) to it:

```
120 FOR I = 1 TO 50
130    S=S+X(I)
140 NEXT I
```

Finally, print the sum:

```
150 PRINT "THE SUM OF ALL THE NUMBERS IS";S
```

Problem: find the average value of X. In other words, find the average of the 50 numbers. Solution: begin by finding the sum—

```
110 S=0
120 FOR I = 1 TO 50
130    S=S+X(I)
140 NEXT I
```

and then divide the sum by 50:

```
150 PRINT "THE AVERAGE IS";S/50
```

Problem: find out whether any of the values of X is 79.4. In other words, find out whether 79.4 is a number in the list. Solution: if X(I) is 79.4, print "YES"—

```
110 FOR I = 1 TO 50
120    IF X(I)=79.4 THEN PRINT "YES, 79.4 IS IN THE LIST": STOP
130 NEXT I
```

otherwise, print "NO":

```
140 PRINT "NO, 79.4 IS NOT IN THE LIST"
```

Problem: in the list of numbers, count how often the number 79.4 appears. Solution: start the counter at zero—

```
110 C=0
```

and increase the counter each time you see the number 79.4:

```
120 FOR I = 1 TO 50
130    IF X(I)=79.4 THEN C=C+1
140 NEXT I
```

Finally, print the counter:

```
150 PRINT "THE NUMBER 79.4 APPEARS";C;"TIMES"
```

Problem: print all the values of X that are negative. In other words, print all the numbers that have minus signs. Solution: begin by announcing your purpose—

```
110 PRINT "HERE ARE THE VALUES THAT ARE NEGATIVE:"
```

and then print the values that are negative; in other words, print each X(I) that's less than 0:

```
120 FOR I = 1 TO 50
130    IF X(I)<0 THEN PRINT X(I)
140 NEXT I
```

Problem: print all the values of X that are "above average". Solution: find the average, and make A the average—

```
110 S=0
120 FOR I = 1 TO 50
130    S=S+X(I)
140 NEXT I
150 A=S/50
```

then announce your purpose:

```
160 PRINT "THE FOLLOWING VALUES ARE ABOVE AVERAGE:"
```

Finally, print the values that are above average; in other words, print each X(I) that's greater than A:

```
170 FOR I = 1 TO 50
180    IF X(I)>A THEN PRINT X(I)
190 NEXT I
```

Problem: find the biggest value of X. In other words, find which of the 50 numbers is the biggest. Solution: let B stand for the biggest number. Begin by tentatively setting B equal to the first number—

```
110 B=X(1)
```

but if another number is bigger than that B, change B:

```
120 FOR I = 1 TO 50
130    IF X(I)>B THEN B=X(I)
140 NEXT I
```

Afterwards, print B:

```
150 PRINT "THE BIGGEST NUMBER IN THE LIST IS";B
```

Problem: find the smallest value of X. In other words, find which of the 50 numbers is the smallest. Solution: let S stand for the smallest number. Begin by tentatively setting S equal to the first number—

```
110 S=X(1)
```

but if another number is smaller than S, change S:

```
120 FOR I = 1 TO 50
130    IF X(I)<S THEN S=X(I)
140 NEXT I
```

Afterwards, print S:

```
150 PRINT "THE SMALLEST NUMBER IN THE LIST IS";S
```

Problem: find out whether the values of X are in strictly increasing order. In other words, find out whether the following statement is true: X(1) is a smaller number than X(2), which is a smaller number than X(3), which is a smaller number than X(4), etc. Solution: if X(I) is *not* smaller than X(I+1), print "NO"—

```
110 FOR I = 1 TO 49
120    IF X(I)>=X(I+1) THEN PRINT "NO, THE LIST IS NOT IN STRICTLY INCREASING OR
DER": STOP
130 NEXT I
```

otherwise, print "YES":

```
140 PRINT "YES, THE LIST IS IN STRICTLY INCREASING ORDER"
```

Multiple Arrays

Suppose your program involves three lists of numbers. Suppose the first line is called A; the second list is called B; and the third list is called X. Suppose A's list contains 18 numbers, B's list contains 57 numbers, and X's list contains just 3 numbers. To say all that, begin your program with this statement:

```
10 DIM A(18), B(57), X(3)
```

Double Subscripts

You can make X be a *table* of numbers, like this:

$$X=\begin{pmatrix} 57 & 8.4 & -6 \\ 1000 & 0 & 7.77 \end{pmatrix}$$

Here's how to make X be that table....

Begin by saying:

```
10 DIM X(2,3)
```

That says X will be a table, having 2 rows and 3 columns.

Then tell the computer what numbers are in X. Type these lines:

```
20 X(1,1)=57
30 X(1,2)=8.4
40 X(1,3)=-6
50 X(2,1)=1000
60 X(2,2)=0
70 X(2,3)=7.77
```

Line 20 says: the number in X's first row and first column is 57. Line 30 says the number in X's first row and second column is 8.4. The remaining lines define the other numbers in X.

If you'd like the computer to print all those numbers, type this:

```
80 MAT PRINT X
```

That means: print all the numbers in X. The computer should print:

```
 57             8.4            -6
 1000           0               7.77
```

If you have a PDP-11 or PDP-20, and want it to print a table, you must put a comma after MAT PRINT, like this:

```
80 MAT PRINT X,
```

If you have a microcomputer that doesn't understand MAT, say this:

```
80 FOR I = 1 TO 2: FOR J = 1 TO 3: PRINT X(I,J),: NEXT: PRINT: NEXT
```

In that program, X is called a *table* or a *doubly subscripted array*.

Multiplication Table

This program prints a multiplication table:

```
10 DIM X(10,4)
20 FOR I = 1 TO 10
30     FOR J = 1 TO 4
40             X(I,J)=I*J
50     NEXT J
60 NEXT I
70 MAT PRINT X
```

Line 10 says X will be a table having 10 rows and 4 columns. Line 40 says the number in row I and column J is I*J; for example, the number in row 3 and column 4 is 12. Lines 20 and 30 make sure line 40 is done for *every* I and J, so *every* entry in the table is defined by multiplication. Line 70 prints the whole table:

```
 1              2              3              4
 2              4              6              8
 3              6              9              12
 4              8              12             16
 5              10             15             20
 6              12             18             24
 7              14             21             28
 8              16             24             32
 9              18             27             36
 10             20             30             40
```

On PDP-11 and PDP-20 computers, put a comma after MAT PRINT:

```
70 MAT PRINT X,
```

On microcomputers, say this instead of MAT PRINT:

```
70 FOR I = 1 TO 10: FOR J = 1 TO 4: PRINT X(I,J),: NEXT: PRINT: NEXT
```

(If your microcomputer's screen is so narrow that it can't fit all four columns, the rightmost columns will be printed strangely.)

Instead of multiplication, you can have addition, subtraction, or division: just change line 40.

Most programmers follow this tradition: the row's number is called I, and the column's number is called J. Line 40 obeys that tradition. Notice I comes before J in the alphabet; I comes before J in X(I,J); and "FOR I" comes before "FOR J" in lines 20-30. If you follow the I-before-J tradition, you'll make fewer errors.

Sums

Suppose you want to analyze this table:

```
  32.7    19.4    31.6    85.1
  -8      402    -61       0
5106       -.2     0      -1.1
  36.9      .04    1       11
 777      666     55.44     2
   1.99     2.99    3.99     4.99
  50       40      30       20
  12       21      12       21
   0     1000       2      500
```

The table has 9 rows and 4 columns; so begin your program by saying:

```
10 DIM X(9,4)
```

Each row of the table becomes a row of the DATA:

```
11 DATA 32.7, 19.4, 31.6, 85.1
12 DATA -8, 402, -61, 0
13 DATA 5106, -.2, 0, -1.1
14 DATA 36.9, .04, 1, 11
15 DATA 777, 666, 55.44, 2
16 DATA 1.99, 2.99, 3.99, 4.99
17 DATA 50, 40, 30, 20
18 DATA 12, 21, 12, 21
19 DATA 0, 1000, 2, 500
```

Make the computer READ the data:

```
20 MAT READ X
```

If you have a microcomputer that doesn't understand MAT, say this instead:

```
20 FOR I = 1 TO 9: FOR J = 1 TO 4: READ X(I,J): NEXT: NEXT
```

To make the computer print the table, say this, on most computers:

```
30 MAT PRINT X
```

Say this, on PDP-11 and PDP-20 computers:

```
30 MAT PRINT X,
```

Say this, on microcomputers:

```
30 FOR I = 1 TO 9: FOR J = 1 TO 4: PRINT X(I,J),: NEXT: PRINT: NEXT
```

Problem: find the sum of all the numbers in the table. Solution: start the sum at 0—

```
100 S=0
```

and then increase the sum, by adding each X(I,J) to it:

```
110 FOR I = 1 TO 9
120     FOR J = 1 TO 4
```

```
130            S=S+X(I,J)
140    NEXT J
150 NEXT I
```

Finally, PRINT the sum:

```
160 PRINT "THE SUM OF ALL THE NUMBERS IS";S
```

The computer will print:

```
THE SUM OF ALL THE NUMBERS IS 8877.84
```

Problem: find the sum of each row. In other words, make the computer print the sum of the numbers in the first row; and then print the sum of the numbers in the second row; and then print the sum of the numbers in the third row; etc. Solution: the general idea is—

```
100 FOR I = 1 TO 9
110    print the sum of row I
120 NEXT I
```

Here are the details:

```
100 FOR I = 1 TO 9
110    S=0
111    FOR J = 1 TO 4
112            S=S+X(I,J)
113    NEXT J
114    PRINT "THE SUM OF ROW";I;"IS";S
115 NEXT I
```

The computer will print:

```
THE SUM OF ROW 1 IS 168.8
THE SUM OF ROW 2 IS 333
THE SUM OF ROW 3 IS 5104.7
etc.
```

Problem: find the sum of each column. In other words, make the computer print the sum of the numbers in the first column; and then print the sum of the numbers in the second column; and then print the sum of the numbers in the third column; etc. Solution: the general idea is—

```
100 FOR J = 1 TO 4
110    print the sum of column J
120 NEXT J
```

Here are the details:

```
100 FOR J = 1 TO 4
110    S=0
111    FOR I = 1 TO 9
112        S=S+X(I,J)
113    NEXT I
114    PRINT "THE SUM OF COLUMN";J;"IS";S
115 NEXT J
```

The computer will print:

```
THE SUM OF COLUMN 1 IS 6008.59
THE SUM OF COLUMN 2 IS 2151.23
THE SUM OF COLUMN 3 IS 75.03
THE SUM OF COLUMN 4 IS 642.99
```

In all the other examples we looked at, "FOR I" comes before "FOR J"; but in this unusual example, "FOR I" comes *after* "FOR J".

String Arrays

You've seen that X can be a list of numbers, or a table of numbers. Similarly, on most computers, X$ can be a list of strings, or a table of strings. For example, you can make X$ be this table—

```
      ("DOG"      "CAT"      "MOUSE"     )
X$=   ("WOOF"     "MEOW"     "SQUEAK"    )
      ("HOTDOG"   "CATSUP"   "MOUSETARD" )
```

by saying this—

```
10 DIM X$(3,3)
20 X$(1,1)="DOG"
30 X$(1,2)="CAT"
40 X$(1,3)="MOUSE"
50 X$(2,1)="WOOF"
etc.
```

or by saying this:

```
10 DIM X$(3,3)
20 DATA DOG,CAT,MOUSE,WOOF,MEOW,SQUEAK,HOTDOG,CATSUP,MOUSETARD
30 MAT READ X$   Micro: FOR I = 1 TO 3: FOR J = 1 TO 3: READ X$(I,J): NEXT: NEXT
```

That works on *most* computers, but not *all* computers. Here are the details....

PDP-11, PDP-20, Honeywell: you can use string lists and tables, without any difficulty.
PDP-10, Univac: you can use string *lists*, but not string *tables*.
Most microcomputers, CDC: you can use string lists and tables; you cannot use MAT with them.
Indecent computers (refer back to the section that explained which computers were decent and indecent): you cannot use string lists or tables.
IBM: you can use string lists and tables; to say that Y$ will be a list of 7 strings, and each of the strings will have 4 characters, say DIM Y$4(7).

The Author

(An Autobiography)

The author ("Russy-poo" Walter) and the modern computer ("stored-program computer") were both conceived in 1946. Nine months later, Russy-poo was hatched; the modern computer took a few years longer. So Russ got a head start. But the computer quickly caught up. Ever since, they've been racing against each other, to see who's smartest. The outcome is still undecided.

The race is close, because Russ and the computer have so much in common. Impartial observers say the computer "acts human", and that Russ's personality is "as dead as a computer".

Yes, Russ is like a computer in many ways. For example, both are Jewish. The father of the modern computer was John von Neumann, a Germanic Jew who fled the Nazis and became a famous U.S. mathematician. The father of Russy-poo Walter was was Henry Walter, a Germanic Jew who fled the Nazis and became a famous U.S. dental salesman: to dentists, he sold teeth, dental chairs, and balloons (for the kids).

To try to get ahead of the computer, Russ got his bachelor's degree in math from Dartmouth in yummy '69, and has sadly remained a bachelor ever since (unless you count the computer he got married to). After Dartmouth, he got an M.A.T. in mathematical education from Harvard. Since he went to Harvard, you know he's a genius. Like most genii, he achieved the high honor of being a junior-high teacher.

After his classes showered him with the Paper Airplane Award, he moved on, to teach at an exclusive private schools (for very exclusive girls) in New York City. ("Exclusive" means everyone can come except you.) After teaching every grade from 2 through 12 (he taught the 2nd graders how to run the computer, and the 12th graders less intellectual things), he fled reality by joining Wesleyan University's math Ph.D. program in Connecticut's Middletown (the middle of Nowhere), where, after a year-and-a-half of highbrow hoopla, he was seduced by a computer, to whom he's now happily married.

The computer he married is a daughter of his competitor. Actually, they're not legally married— she's under age— but she spends most of her life near (or in) his bed; he just "shacks up" with her; when he's away on a trip, he's sometimes tempted to communicate with her via radio, which is how she got the name "radio shack".

After the wedding, he moved with his electrifying "wife" to Northeastern University in Boston (home of the bean and the cod), where he did a hilarious job of teaching in what was lewdly known as the Department of "Graphic Science". After quitting Northeastern and also editorship of *Personal Computing*, the magazine that imitates *Playboy* but replaces the girls by computers, and which goes to 30,000 homes that are full of computers instead of cats, he spends his time now happily losing money by publishing this book.

The young couple returns to Wesleyan in the summers, to give their ever popular (and ever ridiculous) course on *How to Become a Computer Expert in Six Weeks (and Regret It)*, which "comes" to a "climax" when Russ's young wife does a strip tease in front of the students: after the students enjoy a close-up look at her most intimate parts, she gives the students hands-on experience, to teach them a thing or two about computer anatomy.

Index

To program the computer, you must know many words. Here they are. Next to each word, I've indicated the page on which the word is explained thoroughly.

The words fall into three categories. To talk to other programmers, you use "jargon" words; to talk to the computer itself, you use "statement" words and "command" words. You put the "statement" words into your program; you use the "command" words to tell the computer what to do to your program.

Here are all the words. You don't have to look at them now; but later, when you're dealing with the computer, you'll want to refer back to this page, for help.

JARGON

STATEMENTS

COMMANDS

COMMANDS

SECRET PROGRAM LISTING	Name of Program: _____
Name of Programmer: _____	Date Written: _____
Purpose of Program: _____	

Line No.	Statements

PAGE No. _____

SECRET PROGRAM LISTING	Name of Program: _____
Name of Programmer: _____	Date Written: _____
Purpose of Program: _____	

Line No.	Statements

PAGE No. _____

SECRET PROGRAM LISTING	Name of Program: _____
Name of Programmer: _____	Date Written: _____
Purpose of Program: _____	

Line No.	Statements

PAGE No. _____

SECRET PROGRAM LISTING	Name of Program: _____
Name of Programmer: _____	Date Written: _____
Purpose of Program: _____	

Line No.	Statements

PAGE No. _____

SECRET PROGRAM LISTING	Name of Program: _____
Name of Programmer: _____	Date Written: _____
Purpose of Program: _____	

Line No.	Statements

PAGE No. _____

SECRET PROGRAM LISTING	Name of Program: _____
Name of Programmer: _____	Date Written: _____
Purpose of Program: _____	

Line No.	Statements

PAGE No. _____

SECRET PROGRAM LISTING	Name of Program: _____
Name of Programmer: _____	Date Written: _____
Purpose of Program: _____	

Line No.	Statements

PAGE No. _____

SECRET PROGRAM LISTING	Name of Program: _____
Name of Programmer: _____	Date Written: _____
Purpose of Program: _____	

Line No.	Statements

PAGE No. _____

SECRET PROGRAM LISTING	Name of Program: _____
Name of Programmer: _____	Date Written: _____
Purpose of Program: _____	

Line No.	Statements

PAGE No. _____

SECRET PROGRAM LISTING	Name of Program: _____
Name of Programmer: _____	Date Written: _____
Purpose of Program: _____	

Line No.	Statements

PAGE No. _____

SECRET PROGRAM LISTING	Name of Program: _____
Name of Programmer: _____	Date Written: _____
Purpose of Program: _____	

Line No.	Statements

PAGE No. _____

SECRET PROGRAM LISTING	Name of Program: _____
Name of Programmer: _____	Date Written: _____
Purpose of Program: _____	

Line No.	Statements

PAGE No. _____

This volume turns you into a computer expert, quickly and easily. It explains the kind of computer found in most schools, small businesses and homes: the kind that has interactive BASIC. Just curl up in your favorite chair, put this book in your lap and have fun.

In this volume, you'll find four lessons. The first lesson, called "Chat with Your Computer," explains how to deal with computer machinery -- which buttons to press to make it "go" — and trains you to write many kinds of programs. When you finish this lesson, you'll know how to program the computer so that it replaces your typewriter, your photocopy machine and your calculator. How can one computer replace those three other machines? You'll know. And you'll be programming the computer with confidence.

As you read this lesson, you'll want to try everything on a computer. If you bought your own computer — or got permission to use somebody else's — great! If not, don't fret: You can understand the lesson anyway.

Throughout the lesson, you'll see many charts comparing the different computers. Do not try to read every line of every chart! Read about just the computer that you have. If you don't have a computer yet, pretend you have Radio Shack's (which is the simplest).

The second lesson, called "Make Your Computer Think," explains how to make the computer think like a human. The lesson climaxes when you learn how to make the computer imitate your own personality. You'll also learn how to make the computer write clever stories and print letters admitting you to Harvard.

Some programmers believe that every concept in the universe (such as love, hate, Nixon, God and pickles) can be explained by using just those five words. When you finish this lesson, you'll understand those five magic words.

The words are PRINT, INPUT, GO, STOP and IF. At first, you'll wonder, "Explain everything in the whole universe by using just those five words? You must be crazy!" But when you finish the lesson, you won't be so sure — by then you'll have seen hard-nosed examples of how so many things in life can be reduced to those five words. "Ya gotta see it to believe it!"

After the mind-blowing stuff in Lesson 2, Lesson 3 comes back down to earth. You learn how to make the computer solve practical problems about debts, diets, translating French, and other real-world hangups. You get helpful inside hints on how to remove errors from programs, so that the programs work perfectly. You learn how to make the computer print boring tables (in case you have a boss you'd like to bore). Everything good and bad about the everyday practical world of computers comes to light here.

But don't think Lesson 3 remains totally on earth. No way! In fact, this lesson contains the biggest mind-blower of all: The computer can make its own decisions without being told what to do. Yes, the computer, like people, can get inspired, and do something totally "off the wall." This lesson explains how to tell the computer to "get inspired" and make its own decisions — by using "random numbers." Also, this lesson contains the greatest fun. You'll learn how to make the computer play games with you, so that neither you nor the computer can predict who'll win!

Lesson 4 will challenge you so you can give your own brain a whirl. It digs into the toughest parts of programming: loop techniques and subscripts.

Have fun!