

ADAM™

SmartBASIC™

PROGRAMMING WITH ADAM™

A Simple Guide to SmartBASIC™



ADAMTM SmartBASICTM

**PROGRAMMING MANUAL
REVISED EDITION**

WARNING ON CLASS B PRODUCTS

This equipment generates and uses radio frequency energy and if not installed and used properly, that is, in strict accordance with the manufacturer's instructions, may cause interference to radio and television reception. It has been type tested and found to comply with the limits for a Class B computing device in accordance with the specifications in Subpart J of Part 15 of FCC Rules, which are designed to provide reasonable protection against such interference in a residential installation. However, there is no guarantee that interference will not occur in a particular installation. If this equipment does cause interference to radio or television reception, which can be determined by turning the equipment off and on, the user is encouraged to try to correct the interference by one or more of the following measures:

- Reorient the receiving antenna;
- Relocate the computer with respect to the receiver;
- Move the computer away from the receiver;
- Plug the computer into a different outlet so that computer and receiver are on different branch circuits.

If necessary, the user should consult the dealer or an experienced radio/television technician for additional suggestions. The user may find the following booklet prepared by the Federal Communications Commission helpful:

"How to Identify and Resolve Radio/TV Interference Problems".

This booklet is available from the U.S. Government Printing Office, Washington, DC 20402. Stock No. 004-000-00345-4

Warning: This equipment has been certified to comply with the limits for a Class B computing device, pursuant to Subpart J of Part 15 of FCC Rules. Only peripherals (computer input/output devices, terminals, printers, etc.) certified to comply with the Class B limits may be attached to this computer. Operation with non-certified peripherals is likely to result in interference to radio and TV reception.

Use of cables other than the Coleco cables (or equivalent) specified in this manual to connect peripheral equipment like printers, modems or video monitors will invalidate the Federal Communications Commission Certification of your computer and may cause interference levels exceeding the limits established by the FCC for this equipment.

When using any expansion ports for accessory equipment, shielded connecting cable may be required. Contact the accessory manufacturer for further details.

LIMITED NINETY DAY WARRANTY

Coleco warrants to the original consumer purchaser of its ADAM Family Computer System in the United States of America and Canada that the product will be free of defects in material or workmanship for 90 days from the date of purchase under normal in-house use.

Coleco's sole and exclusive liability for defects in material and workmanship shall be limited to repair or replacement at its authorized Coleco Service Station. This warranty does not obligate Coleco to bear the cost of transportation charges in connection with the repair or replacement of defective parts.

This warranty is invalid if the damage or defect is caused by accident, act of God, consumer abuse, unauthorized alteration or repair, vandalism, or misuse.

Any implied warranties arising out of the sale of the ADAM Family Computer System including the implied warranties of merchantability and fitness for a particular purpose are limited to the above 90 day period. Coleco shall in no event be liable for incidental, consequential, contingent, or other damages.

This warranty gives you specific legal rights and you may have other rights which vary from State to State. Some states do not allow the exclusion or limitation of incidental or consequential damages or limitations on how long an implied warranty lasts, so the above limitations or exclusions may not apply to you.

SERVICE POLICY

Please read the Set-Up Manual carefully before using the product. If your ADAM Family Computer System fails to operate properly, please refer to the troubleshooting checklist in the Set-Up Manual. If you cannot correct the malfunction after consulting the troubleshooting checklist, please call Customer Service on Coleco's toll-free hotline: 1-800-842-1225 nationwide. In Canada, 1-800-361-2122. This service is in operation from 8:00 a.m. to 5:00 p.m. Eastern Standard Time, Monday through Friday.

If Customer Service advises you to return your ADAM Family Computer System, please carry it in or return the entire system postage prepaid and insured, in the original box (if possible), with your name, address, proof of the date of purchase, and a brief description of the problem to the Service Station you have been directed to return it to by the toll-free service information. If your unit is found to be factory defective during the Limited Warranty period, it will be repaired or replaced at no cost to you. If the unit is found to have been consumer damaged or abused and therefore not covered by the warranty, then you will be advised, in advance, of repair costs.

If your computer requires service after expiration of the Limited Warranty period, please call Coleco's toll-free service hotline for instructions on how to proceed: 1-800-842-1225 nationwide. In Canada, 1-800-361-2122.

IMPORTANT: SAVE YOUR RECEIPTS SHOWING DATE OF PURCHASE

ACKNOWLEDGEMENTS

We wish to thank the following Coieco employees who were instrumental in assembling this manual:

EDITOR:

Barbara Spear

TECHNICAL CONSULTANTS:

Mark Callahan

Michael DeManche

Portions of this manual are an adaptation of The Secret Guide To Computers written by Russ Walter.

Portions of this manual were written by Barbara Spear and Michael DeManche.

Table of Contents

LESSON 1: CHATTING WITH ADAM Page 7-30

Your first steps: be bold, how to use the keys,
Errors: cancel a character, correct a mistake,
gripes, bloopers.
Fun: spaces, semicolons, rearranging, GOTO,
conveniences, leaving.
Math: arithmetic, E, range, order, parentheses.

LESSON 2: ADAM THINKS Page 31-53

Variables: numeric variables, when to use
variables, string variables.
Input: string input, numeric input.
Digital Data Packs: SAVE, CATALOG, LOAD
Logic: STOP, colons, IF.

LESSON 3: MASTER YOUR COMPUTER Page 55-88

Helpful hints: reminders, debugging, editing,
pause.
Paired statements: DATA. . .READ, FOR. . . NEXT.
Popular features: random numbers, zones.

LESSON 4: TACKLE THE TOUGH STUFF Page 89-116

Loop techniques: incrementing, decrementing,
counting, summing.
Subscripts: single subscripts, double subscripts,
string arrays, graphics.

APPENDIX A: REFERENCE SECTION

APPENDIX B: ADVANCED REFERENCE SECTION

APPENDIX C: COMPENDIUM OF USEFUL PROGRAMMING INFORMATION

THE FOUR LESSONS

This book has been divided into four lessons. Each lesson is about 20 pages long, and can easily be finished in a brief sitting. Reference information for those who already know BASIC can be found in appendices A, B, and C.

LESSON 1: CHATTING WITH ADAM

This first lesson explains which buttons to press, how to make it "go", and trains you to write many kinds of programs. When you finish this lesson, you'll know how to make ADAM replace your typewriter, your photocopier, and your calculator. How can one computer replace all those three other machines? You'll know. And you'll be programming the computer with confidence.

LESSON 2: ADAM THINKS

This lesson explains how ADAM can 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, politicians, Santa Claus, and non-dairy coffee creamer) can be explained by using just five words. The words are: PRINT, INPUT, GO, END, and IF. When you finish this lesson, you'll understand those five magic words.

LESSON 3: MASTER YOUR COMPUTER

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 hang-ups. You get helpful inside hints on how to remove errors from programs, so that the programs work perfectly. You even learn how to make the computer print boring tables (in case you have a boss you'd like to bore).

But don't think Lesson 3 remains totally on earth. No way! In fact, this lesson contains the biggest mind-blower of all: ADAM can make its own decisions. This lesson explains how to tell the computer to make its own decisions. . . by using "random numbers".

You'll learn how to make ADAM play games with you, so that neither you nor the computer can predict who'll win!

LESSON 4: TACKLE THE TOUGH STUFF

This lesson will challenge you, so you can give your own brain a whirl. It digs into the toughest parts of programming: "Loop techniques", "subscripts", and graphics. It also shows you how incredibly powerful your ADAM computer really is. Have fun!

MODEL PROGRAMS

Each lesson contains six model programs. Here they are. You don't have to look at them yet. Later, when you're writing your own programs, you may want to refer back to these pages for help.

LESSON 1: CHATTING WITH ADAM

Printing

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

The computer prints:

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

Semicolon

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

The computer prints FAT and HER on the same line:

```
FATHER
```

Embedded semicolon

```
1 PRINT "THIN"; "KING"
```

The computer prints THIN and KING on the same line:

```
THINKING
```

Goto

```
10 PRINT "CAT"
20 PRINT "DOG"
30 GOTO 10
```

The computer prints CAT and DOG repeatedly.

Arithmetic

```
10 PRINT 1+2
```

The computer computes 1+2 and prints the answer:

```
3
```

Order of operations

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

The computer starts with 2, then adds three fours. It prints:

```
14
```


LESSON 2: ADAM THINKS

Variable

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

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

49

Input

```
10 INPUT "WHAT IS YOUR NAME?";N$
20 PRINT "YOU HAVE A NICE
NAME, ";N$
```

Line 10 makes ADAM ask-- WHAT IS YOUR NAME? then wait for you to answer; your answer is called N\$. If you answer ALBERT, line 20 prints:

YOU HAVE A NICE NAME, ALBERT

Colon

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

The computer prints:

12

String variable

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

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

DOWN

Stop

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

The computer prints just:

BUBBLE

IF

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

Line 10 asks if you're male or female. After you answer, the computer prints an appropriate retort.

LESSON 3: MASTER YOUR COMPUTER

Data

```
10 DATA MEAT,POTATOES,  
20 READ A$  
30 PRINT A$  
40 GOTO 20
```

The computer
prints:

MEAT
POTATOES
LETTUCE
OUT OF DATA

For

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

The computer
prints every number
from 1 to 100.

Def_Fn

```
10 DEF FN D(N)=INT(1+N*RND(1))  
20 PRINT FN D(5)
```

The computer prints
1 or 2 or 3 or 4 or 5.

Restore

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

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

Step

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

The computer prints a
rocket countdown:

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

Comma

```
10 PRINT  
"THIN","KING"
```

The computer
prints THIN in the
first zone, and
KING in the
second:

THIN

KING

LESSON 4: TACKLE THE TOUGH STUFF

Incrementing

```
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

Summing

```
10 S=0
20 PRINT "NOW THE SUM IS";S
30 INPUT "WHAT NUMBER DO YOU
    WANT ADDED INTO THE SUM? ";X
40 S=S+X
50 GOTO 20
```

The computer imitates an adding machine

Read

```
10 DIM X(3)
20 DATA 57.2, -8.3, 476
30 FOR I = 1 TO 3
40 READ X(I)
50 PRINT X(I)
60 NEXT I
```

Line 10 says X is a list of 3 numbers. Line 30 reads them from the data. Line 50 prints

57.2
-8.3
476

Counting

```
10 C=3
20 PRINT C
30 C=C+1
40 GOTO 20
```

The computer counts, starting at 3. It prints: 3, 4, 5, 6, etc.

Subscripts

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

Line 10 says X is a list of 3 numbers. Lines 20-40 define them. Lines 50-70 prints them:

57.2
-8.3
476

Double subscripts

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

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

LESSON 1

CHATting WITH ADAM

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. ADAM is invincible! So go ahead and experiment. If it doesn't like what you say, it will gripe at you, but so what? You wanted a computer that talks to you, right?

Furthermore, anyone is bound to make a mistake when they first use their computer. Fortunately, ADAM tells you before you've gone too far, and often tells you exactly what you've done wrong. So, whenever you have trouble, laugh about it, and say, "Oh boy! Here we go again!" Then get some help.

If you have friends that work with computers or know of a user group, 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, so play along with the computer people, boost their egos, and they'll help you, too. 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. If you're taking a computer course, be sure you get your money's worth and ask your teacher and coworkers questions, questions, questions!

If you're using ADAM on a personal basis and find you need help, you can always call us at the number listed in and the SERVICE POLICY SECTION.

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

How To Use The Keyboard

This section outlines the uses of the keys.

Numbers

The top row of keys contains the numbers. Don't confuse the number one with the letter l; 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 will appear on your keyboard and screen with a slash through it.

The SHIFT key

One of your keys has the number four with a dollar sign above it. If you press this key, you'll type a 4. But, if you hold down the SHIFT key, and then press the key with the 4 and the dollar sign, you'll find that you're typing the dollar sign this time.

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. On a single symbol key, hold the shift key down to get a capital letter.

About the SHIFT key. . . beginners often make two boo-boos. The first boo-boo is to forget to press it when you want an upper character on a key. 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. ADAM also has a LOCK key. Each time the LOCK key is pressed, it switches between upper and lower case characters.

The RETURN key

The most important key is the RETURN key. 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.

Loading SmartBASIC

After you've stared at the keyboard, you really should get into SmartBASIC. Here's how:

- Turn the computer on.
- Place SmartBASIC digital data pack into Drive.
- Hit the reset key.
- Wait for several seconds.

The computer will print a bracket on the screen "]", as well as "Coleco SmartBASIC VI.0 at the top of the screen. You are now ready to begin talking to your computer.

NOTE: Do not turn the computer on or off with a digital data pack in the drive.

Programming

Now that you've gotten into SmartBASIC, you're ready to program the computer!

The bracket on the left side of your screen means that ADAM is waiting for a command. Don't type a command until you see the bracket and the blinking cursor. 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 and to forget your old program. After you type the word NEW, press the RETURN key. The computer will print a bracket on the screen.

Step 2: type your program A program is a list of numbered commands. You can program ADAM in all upper case, upper and lower, or all lower case; it's your choice. 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 above program must say PRINT; you must type the word PRINT. Every line of the above 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 LIST. That tells the computer to print your program on the screen so you can check to see if you've typed the program the way you wanted to.

If you type your program out in lower case, and LIST it, you'll find that ADAM converted it to upper case automatically. The only time ADAM will not convert to upper case automatically is if your lower case material is the name of a variable or within quotes in a PRINT statement. Upper case and lower case variable names many look different to you, but ADAM sees them as the same.

Step 4: 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 print a bracket on the screen.

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 programs is called a programmer. Congratulations: you're a programmer!

Please note that, as they stand, the sample programs in this book, will not print out on your printer. Unless you type in the instruction to do so, they will appear on your screen only. To print them out, use the command PR#1. When you are all through printing, use the command PR#0 to send your output back to the screen. For more information on PR#, see the Reference Section at the back of this book (Appendix A).

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, first, turn on the printer with PR#1. Then 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.

You type:

RUN

The computer types:

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

You type:

RUN

The computer types:

SPRING HAS
SPRUNG.
etc.

If you despise 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 flubbed-up business letter.

EDITING

How To Correct a Mistake

If you make a mistake, you can correct it in many ways:

If you notice the mistake before you press the RETURN key, use the BACKSPACE key or left arrow to move the cursor under your mistake. Retype the line from that point.

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 PRINT "I LOVE YOU"
```

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

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

ADAM will tell you about the error by saying ILLEGAL COMMAND. 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"
```


Another way to make corrections is to use the onscreen editing capabilities. (Note: the SmartWRITER editing commands do not work in SmartBASIC.) To use onscreen editing, type "LIST" followed by the number of the line you want to change, then press RETURN. Use the arrow keys to move to the beginning of the line. Use the right arrow key to move the cursor to your mistake. Type your correction. When the line is the way you want it, use the right arrow key to move past the last character. Press RETURN, then check to see if the computer accepted the changes.

To make you life easier, there are control characters that can help you make changes and corrections to your program. While holding the CONTROL key down, press the appropriate key.

CONTROL X "Forgets" the line of
characters being typed

CONTROL L Clears the screen

CONTROL O Deletes the character directly
above the cursor

CONTROL H Backspaces over characters
without deleting them

CONTROL N Inserts spaces to the left of
the cursor

CONTROL P Sends all printable characters
from the screen to the printer
and prints them

CONTROL ← Passes over letters without
"forgetting" them

CONTROL → Passes over letters without
"remembering" them

Using the CONTROL key in combination with the arrow keys can make inserting easier when the line is full.

How to Delete a Line:

To delete a single line, type the line number followed by RETURN.

To delete more than one line, see the DEL command in the Reference Section.

GRIPES

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

```
1 PRINT "ROSES ARE RED"
```

ADAM will gripe at you:

ILLEGAL COMMAND

and a caret (^) will appear to point out where in the line your error lies.

Your next job is to correct your error. To do that, just retype line 1 correctly.

You type this (error):

```
1 PRINT "ROSES  
ARE RED"
```

The computer gripes at you:

ILLEGAL COMMAND

You type this correction

```
1 PRINT "ROSES  
ARE RED"
```

along with the rest of the poem:

```
2 PRINT "CABBAGE  
IS GREEN"  
3 PRINT "MY FACE  
IS FUNNY"  
4 PRINT "BUT  
YOURS IS A  
SCREAM"  
RUN
```

The computer recites:

```
ROSES ARE RED  
CABBAGE IS GREEN  
MY FACE IS FUNNY  
BUT YOURS IS A  
SCREAM
```


Common Bloopers

If you're like most beginners, you'll make these mistakes:

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 the letter l instead of the number one.

You type something (such as RUN), 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 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.

Quotation marks

You must put quotation marks around a data string, if the string contains a comma or a colon.

As a general rule of thumb--when in doubt, or whenever a data statement doesn't seem to be working, insert quotation marks. They can't hurt.

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

You have the ability to compose and print out letters in both SmartWRITER and SmartBASIC modes. Here's how to do it in SmartBASIC:

DEAR JOAN,

THANK YOU FOR THE BEAUTIFUL
NECKTIE. JUST ONE PROBLEM--
I HATE NECKTIES!

LOVE,

TURTLENECK-FRED

This program prints it:

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

In the program, each line (except line 2) 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 with no spaces between the items; so the computer will print HER next to FAT, like this:

FATHER

This program shows how an underweight philosophical king spends his time:

```
1 PRINT "THIN"; "KING"
```

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

THINKING

Rearranging Your Program

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

```
2 PRINT "LOOKING"  
1 PRINT "HERE'S"  
4 PRINT "YOU,"  
3 PRINT "AT"  
5 PRINT "KID"
```

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

```
1 PRINT "HERE'S"  
2 PRINT "LOOKING"  
3 PRINT "AT"  
4 PRINT "YOU,"  
5 PRINT "KID"
```

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

You type this program:

```
2 PRINT "LOOKING"  
1 PRINT "HERE'S"  
4 PRINT "YOU,"  
3 PRINT "AT"  
5 PRINT "KID"
```

You type this command:

LIST

The computer types the program in increasing order:

```
1 PRINT "HERE'S"  
2 PRINT "LOOKING"  
3 PRINT "AT"  
4 PRINT "YOU,"  
5 PRINT "KID"
```

If you type RUN, the computer will print:

```
HERE'S  
LOOKING  
AT  
YOU,  
KID
```

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!

Sore Feet

Type this, and see what happens:

```
NEW  
10 PRINT "MY"  
90 PRINT "TRUCK"  
32 PRINT "SNEEZE"  
50 PRINT "TOE"  
32 PRINT "NEED A"  
70  
80 PRINT "HAT"  
80  
20 PRINT "SORE FEET"
```

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 "NEED A".

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 "MY"  
20 PRINT "SORE FEET"  
32 PRINT "NEED A"  
50 PRINT "TOE"  
90 PRINT "TRUCK"
```

If you type RUN, the computer will print:

```
MY  
SORE FEET  
NEED A  
TOE  
TRUCK
```

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, . . .

GOTO

This program makes the computer go crazy:

```
10 PRINT "CAT"  
20 PRINT "DOG"  
30 GOTO 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 DOG again, and comes back to line 30 again, which makes it go back to line 10 again, print CAT and DOG again, then jump back again. . . the computer will print the words CAT and DOG again and again, like this:

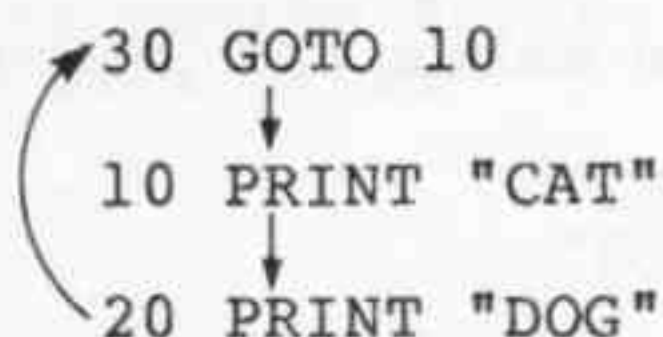
```
CAT  
DOG  
CAT  
DOG  
CAT  
DOG  
CAT  
DOG  
etc.
```

If you use the PR#1 command, the computer will try to print on your printer the words CAT and DOG again and again, forever. Try running that program; you'll have fun watching the computer go crazy. On your screen, you'll see that the computer is printing the words CAT and DOG at the same time the printer is.

To stop this madness, you must give the computer some sort of a jolt that will put it out of its misery. This is called aborting your program. Here's how:

Hold down the CONTROL key; and tap the C key. Or, if ADAM is waiting for input from you when you decide to abort, hold down the CONTROL key, tap the C, and press RETURN.

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 abort the program.

In that program, you typed just three lines; but since the third line said to GOTO the beginning, the computer does an infinite loop. By saying GOTO, 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 GOTO 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 abort your program.

Love

This program's for lovers:

```
10 PRINT "LOVE"  
20 GOTO 10
```

The computer will print LOVE repeatedly, like
this:

```
LOVE  
LOVE  
LOVE  
LOVE  
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 GOTO 10
```


Kiss and tell

Here's a poor, innocent program:

```
10 PRINT "YOUR SISTER"  
20 PRINT "DISLIKES ANYONE WHO"  
30 PRINT "KISSES AND TELLS"
```

It makes the computer print:

```
YOUR SISTER  
DISLIKES ANYONE WHO  
KISSES AND TELLS
```

To make the program more diabolical, insert line 15, by typing it at the bottom of your program and LISTing the program out to see your line re-shuffled into the correct position.

```
10 PRINT "YOUR SISTER"  
15 GOTO 30  
20 PRINT "DISLIKES ANYONE WHO"  
30 PRINT "KISSES AND TELLS"
```

The computer does line 10, which prints YOUR SISTER. Then it does line 15, which makes it skip ahead to line 30, which prints KISSES AND TELLS. So altogether, the computer prints:

```
YOUR SISTER  
KISSES AND TELLS
```

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

Here's a short-cut you should know about. You can get your computer to PRINT without placing a line number in front of the words:

```
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. This short-cut is called immediate mode.

Leaving the Computer

When you're finished using your computer and want to walk away, here's what to do. . . OPEN THE DOOR TO YOUR DIGITAL DATA PACK DRIVE AND TURN EVERYTHING OFF. Now wasn't that easy?

Arithmetic

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

```
10 PRINT 2+2
```

The computer prints:

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

```
10 PRINT 2*6
```

The computer will print:

12

To divide, use the slash beneath the question mark. 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 8516730000.79, or some similar number; what the computer tells you is just an approximation.)

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'll get:

.000926

You won't have to worry much about E notation; the computer uses it only if an answer is very, very large or very small. (ADAM uses E notation only if the answer is bigger than 9999999999 or tinier than .01.) But when the computer does use E notation, remember to move the decimal point!

Range

The highest number ADAM can handle is 1E38 (which is 1 followed by 38 zeros, which is 100). If you try to go much higher, the computer will say OVERFLOW ERROR.

[illegible]

Order of Operations

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

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

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.

ADAM follows 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:

```
2+3*4
    12
```

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:

```
10-2*3+72/9*5
    6    40
```


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

Parentheses

You can use parentheses to change the order by which ADAM does arithmetic. For example, $5*(1+1)$ means $5*2$, which is 10. You can put parentheses inside parentheses: $10-(5*(1+1))$ means $10-(5*2)$, which is $10-10$, which is 0.

LESSON 2

ADAM THINKS

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**. A variable name can be up to 121 letters or numbers, but ADAM only pays attention to the first two. Therefore, APPLE and APE would be understood by ADAM to be the same variable. A variable name must begin with a letter, not a number. Commands and statements cannot be used as variable names. 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.

You may use the statement LET if you like, or just leave it off. With ADAM, the use of LET is optional.

```
LET X=47
or
X=47
```

Further examples:

```
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.

```
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 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 are only allowed to have one variable:

P=M+2 (CORRECT) P-M=2 (WRONG) 2=P-M (WRONG)

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 of 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 to 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 of B; "B=A" changes the value of B but not of A. Compare these programs:

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

```
10 A=1
20 B=7
30 B=A
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 assign a value to a numeric variable, the computer assumes it's zero:

```
10 PRINT R
```

Since R hasn't had a value assigned, the computer 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 unassigned, 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 these six calculations, you could run this program:

```
10 PRINT 1297.43*2, 1297.43*.05, 1297.43+483.19,  
20 PRINT 1297.43/37, 8598.61-1297.43, 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$="DOWN"  
15 PRINT  
20 PRINT G$
```

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.

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 assign characters to 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 assign characters to a string variable, the computer assumes it's blank.

```
10 PRINT F$
```

Since nothing has been assigned to F\$, line 10 makes the computer print a line that says nothing; the line printed will be blank.

String Input

Humans ask questions. So if you want to make the computer work like 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$
15 PRINT
20 PRINT "YOU HAVE A NICE 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\$. For example, if you answer ALBERT, then N\$ is ALBERT. Line 20 makes the computer print: (Be sure to hit the RETURN key to let ADAM know you have finished your input.)

YOU HAVE A NICE NAME, ALBERT

Here's the whole conversation:

(In these lessons, words to be typed in by you will be in bold print.)

You tell ADAM to run:
ADAM asks for your name:

ADAM comments:

```
RUN
WHAT IS YOUR
NAME? ALBERT
YOU HAVE A NICE
NAME, ALBERT
```

If you try that example, be careful! When you type in line 10, which says INPUT, make sure you type the two quotation marks and the semicolon and the question mark. If you have a string of variables following INPUT, then the question mark

will automatically be inserted for you. But if you have a string within quotes, it will not.

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

You tell ADAM to run:	RUN
ADAM asks for your name:	WHAT IS YOUR NAME? MARK
ADAM comments:	YOU HAVE A NICE NAME, MARK

College admissions

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

```
10 INPUT "WHICH COLLEGE WOULD YOU LIKE TO ATTEND?"
   ;C$
16 PRINT
17 PRINT
18 PRINT
20 PRINT "CONGRATULATIONS!"
25 PRINT
30 PRINT "YOU HAVE JUST BEEN ADMITTED TO ";C$
40 PRINT "IT FITS YOUR PERSONALITY"
50 PRINT "I HOPE YOU ATTEND ";C$
55 PRINT
56 PRINT
60 PRINT "                RESPECTFULLY YOURS"
61 PRINT
62 PRINT
63 PRINT
70 PRINT "                THE DEAN OF ADMISSIONS"
```

Line 10 makes the computer ask WHICH COLLEGE WOULD YOU LIKE TO ATTEND?" 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:

WHICH COLLEGE WOULD YOU LIKE TO ATTEND? HARVARD

CONGRATULATIONS!

YOU HAVE JUST BEEN ADMITTED TO HARVARD
IT FITS YOUR PERSONALITY
I HOPE YOU ATTEND HARVARD

RESPECTFULLY YOURS

THE DEAN OF ADMISSIONS

This program consists of three parts:

1. The computer begins by asking you a question (WHICH COLLEGE WOULD YOU LIKE TO ATTEND?) 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 PERSON NAMED

your name

WHO HAD A FRIEND NAMED -----
friend's name

----- wanted
your name

to -----
verb friend's name

BUT ----- DIDN'T WANT
friend's name
TO -----
verb your name

WILL -----
your name verb friend's name

WILL -----
friend's name verb 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:


```

10 INPUT "WHAT'S YOUR NAME? ";Y$
15 PRINT
20 INPUT "WHAT'S YOUR FRIEND'S NAME? ";F$
25 PRINT
30 INPUT "IN 1 WORD, SAY SOMETHING YOU CAN DO TO
YOUR FRIEND ";V$
35 PRINT
40 PRINT "HERE'S MY STORY...."
45 PRINT
50 PRINT "ONCE UPON A TIME, THERE WAS A PERSON
NAMED ";Y$
55 PRINT
60 PRINT "WHO HAD A FRIEND NAMED ";F$
65 PRINT
70 PRINT Y$;" WANTED TO ";V$;" ";F$
75 PRINT
80 PRINT "BUT ";F$;" DIDN'T WANT TO ";V$;" ";Y$
85 PRINT
90 PRINT "WILL ";Y$;" ";V$;" ";F$
95 PRINT
100 PRINT "WILL ";F$;" ";V$;" ";Y$
105 PRINT
110 PRINT "TO FIND OUT, COME BACK AND SEE THE NEXT
EXCITING EPISODE "
115 PRINT
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 PERSON 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? SNOW WHITE
 WHAT'S YOUR FRIEND'S NAME? GRUMPY
 IN 1 WORD, SAY SOMETHING YOU CAN DO TO YOUR
 FRIEND? TICKLE
 HERE'S MY STORY....
 ONCE UPON A TIME, THERE WAS A PERSON NAMED SNOW
 WHITE
 WHO HAD A FRIEND NAMED GRUMPY
 SNOW WHITE WANTED TO TICKLE GRUMPY
 BUT GRUMPY DIDN'T WANT TO TICKLE SNOW WHITE
 WILL SNOW WHITE TICKLE GRUMPY
 WILL GRUMPY TICKLE SNOW WHITE
 TO FIND OUT, COME BACK AND SEE THE NEXT EXCITING
 EPISODE
 OF SNOW WHITE AND GRUMPY

Try it: put in your own name, the name of a friend, and an appropriate verb.

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 CITY ";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
   (BAKED,FRIED,ETC) ";S$
100 PRINT
110 PRINT "CONGRATULATIONS, ";Y$
115 PRINT
120 PRINT "YOU'VE WON THE BEAUTY CONTEST, BECAUSE
   OF YOUR EXTREMELY ATTRACTIVE ";B$
125 PRINT
130 PRINT "YOUR PRIZE IS A ";C$;" ";J$
135 PRINT
140 PRINT "PLUS A TRIP TO ";P$;" WITH YOUR FRIEND
   ";F$
145 PRINT
150 PRINT "PLUS...AND THIS IS THE BEST PART OF
   ALL..."
155 PRINT
  
```



```
160 PRINT "DINNER FOR THE TWO OF YOU AT ";A$;"'S
RESTAURANT"
165 PRINT
170 PRINT "WHERE ";A$;" WILL GIVE YOU ALL THE
";S$;" ";D$;" YOU CAN EAT"
175 PRINT
180 PRINT "CONGRATULATIONS ";Y$;"...TODAY'S YOUR
LUCKY DAY..."
185 PRINT
190 PRINT "NOW EVERYBODY WANTS TO KISS YOUR
AWARD-WINNING ";B$
```

Want to see a sample run? OK:

```
WHAT'S YOUR NAME? ARLO
WHAT'S YOUR FRIEND'S NAME? OBIE
WHAT'S THE NAME OF ANOTHER FRIEND? ALICE
NAME A COLOR? RED
NAME A CITY? WEST STOCKBRIDGE
NAME A FOOD? TURKEY
NAME AN OBJECT? GARBAGE BAG
NAME A PART OF THE BODY? TOENAIL
NAME A STYLE OF COOKING (BAKED,FRIED,ETC)? ROAST
```

```
CONGRATULATIONS, ARLO
YOU'VE WON THE BEAUTY CONTEST, BECAUSE OF YOUR
EXTREMELY ATTRACTIVE TOENAIL
YOUR PRIZE IS A RED GARBAGE BAG
PLUS A TRIP TO WEST STOCKBRIDGE WITH YOUR FRIEND
OBIE
PLUS...AND THIS IS THE BEST PART OF ALL...
DINNER FOR THE TWO OF YOU AT ALICE'S RESTAURANT
WHERE ALICE WILL GIVE YOU ALL THE ROAST TURKEY YOU
CAN EAT
CONGRATULATIONS ARLO...TODAY'S YOUR LUCKY DAY...
NOW EVERYBODY WANTS TO KISS YOUR AWARD-WINNING
TOENAIL
```

Bills

If you're a nasty bill collector, 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$;"", "
125 PRINT
130 PRINT "  YOU STILL HAVEN'T PAID THE BILL."
140 PRINT "IF YOU DON'T PAY IT SOON, ";F$;"", "
150 PRINT "I'LL COME AND VISIT YOU IN ";C$
160 PRINT "AND PERSONALLY PITCH A TENT "
170 PRINT "ON YOUR DOORSTEP UNTIL YOU DO."
175 PRINT
180 PRINT "                YOURS TRULY,"
185 PRINT
190 PRINT "                YOUR CREDITOR"
```

Can you figure out what this 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 INPUT "IN WHAT YEAR WERE YOU BORN? ";Y
30 PRINT "IN THE YEAR 2000, YOU WILL BE
";2000-Y;"."
```

Here's a sample run:

I WILL PREDICT WHAT WILL HAPPEN TO YOU IN THE YEAR
2000!

IN WHAT YEAR WERE YOU BORN? 1954
IN THE YEAR 2000, YOU WILL BE 46.

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
15 PRINT
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
35 PRINT
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
```

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
```

Trying 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 CELSIUS? ";C
20 PRINT C;" DEGREES CELSIUS = ";C*1.8+32;"
DEGREES FAHRENHEIT"
```

Here's a sample run:

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

Digital Data Packs

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

To store the program longer, copy it onto the computer's digital data pack. Once you've copied the program onto a digital data pack, it will remain on the digital data pack, even if you type NEW or turn off the power. It will stay on the digital data pack permanently, unless you rename, delete, save over it, or initialize the data packs.

To copy a program onto the digital data pack, 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 a digital data pack:

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

To name that program "JOE", and copy it onto the digital data pack, type this:

```
NEW
10 PRINT "MY DAD"
20 PRINT "IS SAD"
SAVE JOE
```


Instead of JOE, you can invent a different name, as long as it doesn't exceed 10 characters. In file names, ADAM knows the difference between capital and small letters. For example, "joe", "JOE", and "Joe" would be treated as three different files.

To prove that your saved program is on the digital data pack, type this:

CATALOG

That makes the computer print a catalog of everything you've stored on your digital data pack. The catalog will list the names of all the programs you've stored. And one of the names you'll see will be JOE. See the Reference Section at the back of this book for a full explanation of CATALOG.

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

LOAD JOE

That makes the computer copy JOE from the digital data pack to the main memory. Then you can type LIST or RUN, and the computer will list or run JOE.

If you ever want to erase JOE from the digital data pack, type this:

DELETE JOE

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

SAVE JOE

That makes the computer replace JOE with your new version.

Be aware that SmartBASIC files and ADAM Word Processing files are compatible. In other words, you can take a SmartBASIC file that you've SAVED and READ it into the SmartWRITER word processor as a regular document for editing; but all your PRINTs will show up as "?". You can use either PRINT or ?. Because your work is not checked for syntax errors when you code or edit using word processing this option is not recommended. When you LOAD a program that was created or edited using SmartWRITER, any lines having syntax errors will be rejected.

To keep your files from being deleted accidentally, here's what to do:

LOCK JOE

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, press RETURN and it'll print a bracket. 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

Colons are useful for putting 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.

If/Then

Let's make ADAM interrogate a human. Let's make it 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 NOT SURE or BOTH or YES, let's make the computer say:

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

This will force the human to answer the question correctly. This program does it:

```
10 INPUT "ARE YOU MALE OR FEMALE? ";A$
20 IF A$="MALE" THEN PRINT "SO IS FRANKENSTEIN":
END
30 IF A$="FEMALE" THEN PRINT "SO IS MARY POPPINS":
END
40 PRINT "PLEASE SAY 'MALE' OR 'FEMALE'": GOTO 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. This is one instance where the difference between upper and lower case letters matters.

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$="MALE" THEN PRINT "SO IS
FRANKENSTEIN":END
30 IF A$="FEMALE" THEN PRINT "SO IS MARY POPPINS":
GOTO 100
40 PRINT "PLEASE SAY 'MALE' OR 'FEMALE'": GOTO 10
100 INPUT "DO YOU LIKE HER? ";B$
110 IF B$="YES" THEN PRINT "I LIKE HER TOO--SHE IS
MY MOTHER": END
120 IF B$="NO" THEN PRINT "NEITHER DO I--SHE STILL
OWES ME A DIME": END
130 PRINT "PLEASE SAY 'YES' OR 'NO'": GOTO 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?

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$="YES" THEN PRINT "THAT'S SWELL": END
30 IF A$="NO" THEN PRINT "GO JUMP IN A LAKE": END
40 PRINT "PLEASE SAY 'YES' OR 'NO'": GOTO 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.

Here's a program that 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$="YES" THEN PRINT "THANKS. I'LL BE THERE
AT 5 P.M.": END
30 IF A$="NO" THEN PRINT "HUMPH! I HAVE BETTER
THINGS TO DO, ANYWAY.": END
40 PRINT "PLEASE SAY 'YES' OR 'NO'": GOTO 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 you.

Another sixth grade girl wrote a program to test your honesty:

```
10 PRINT "KDJFUEOEE*^&^%KLO*&@!@>]_"
20 PRINT "LLLL]__))*&(~~++?:LJSF,, "
30 PRINT "JSUEU2JFW098B<,,./[???=-["
40 INPUT "DO YOU UNDERSTAND WHAT I SAID? ";A$
50 IF A$="NO" THEN PRINT "NEITHER DO I": END
60 IF A$="YES" THEN GOTO 80
70 PRINT "PLEASE SAY 'YES' OR 'NO' ": GOTO 10
80 PRINT "IT DOESN'T MEAN ANYTHING!! I'M A
COMPUTER, YOU KNOW! YOU CAN'T FOOL ME!"
```

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 confess that it was beyond understanding. But if you pretend that you understand the nonsense, and answer YES, the computer will scold you for trying to put one over on it.

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": END
40 IF A=4 THEN PRINT "NO, 2 AND 2 IS 22": END
50 IF A=22 THEN 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$="FINE" THEN PRINT "THAT'S GOOD": END  
60 IF F$="AWFUL" THEN PRINT "TOO BAD": END  
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 AWFUL, line 60 makes the computer say TOO BAD. If you say neither FINE nor AWFUL, 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 the program could say:

```
10 PRINT "HELLO"
```

or--

```
10 PRINT "HI"
```

or, if you're from Texas--

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

or, if you're cool--

```
10 PRINT "HEY, 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 nasty--

```
10 PRINT "GO AWAY!  CAN'T YOU SEE I'M SLEEPING?!"
```

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...THEN 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...THEN and GOTO statements.

See how human you can make your ADAM 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

IF A\$="MALE"

IF A=4

IF A<4

IF A<=4

IF A>4

IF A>=4

IF A<>4

IF A\$<"MALE"

IF A\$>"MALE"

Meaning

If A\$ is "MALE"

If A is 4

If A is less than 4

If A is less than or
equal to 4

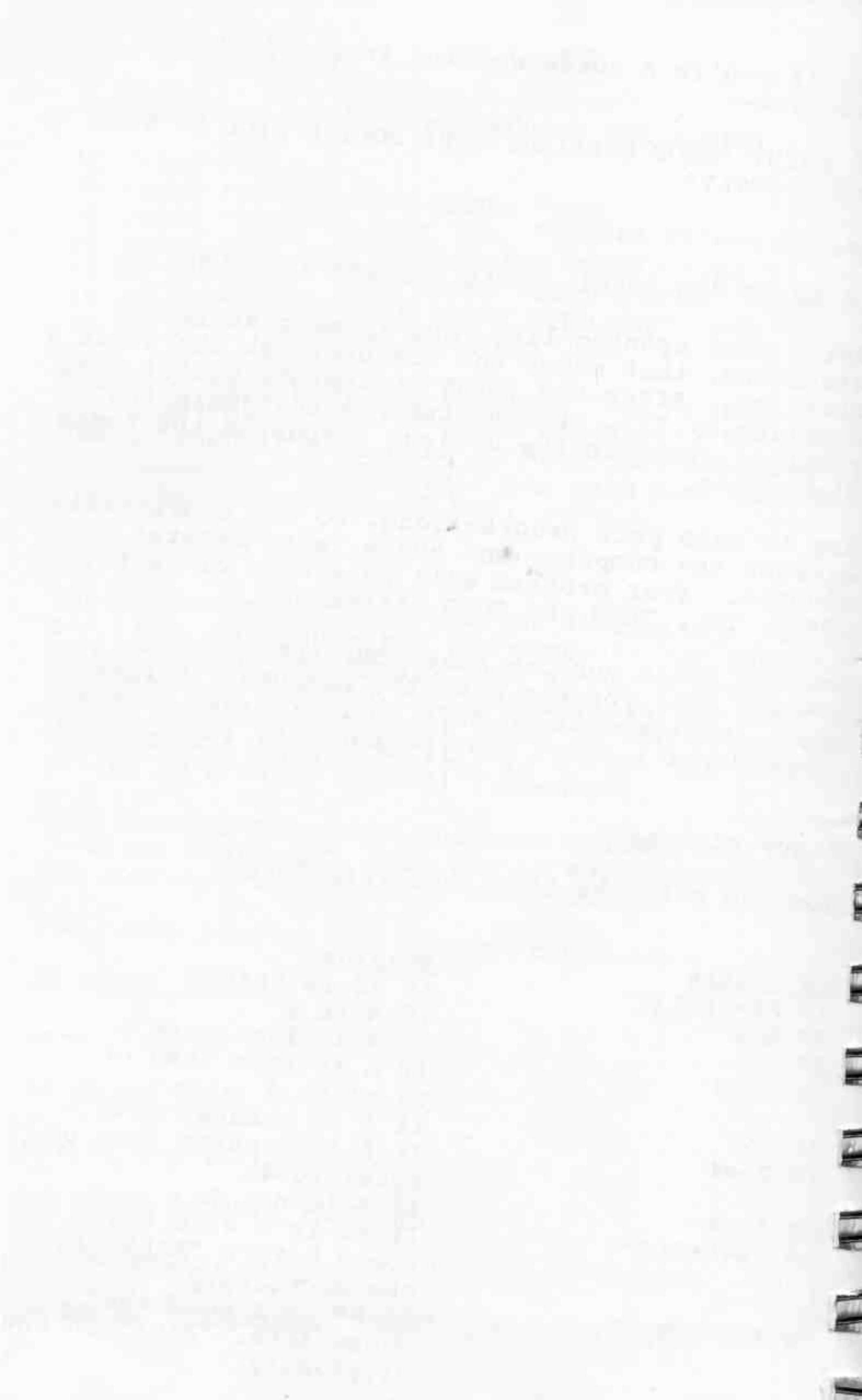
If A is greater than 4

If A is greater than or
equal to 4

If A is not 4

If A\$ is a word that
comes before "MALE" in
the dictionary

If A\$ is a word that
comes after "MALE" in the
dictionary



LESSON 3

MASTER YOUR COMPUTER

Reminders

- If you get lost in your program, type this command:

LIST

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

A\$="JERK"
(CORRECT)

A\$=JERK
(WRONG)

A="JERK"
(WRONG)

- To make your computer execute your program again and again, automatically, add an extra line at the bottom of your program. That extra line should tell the computer to GOTO the top line.
- To say "not equal to", say "less than or greater than", like this: <>.
- Remember that IF is always paired with THEN or GOTO.

The only way to become a computer expert is to try writing your own programs, so go to it!

Debugging

An error in a program is called a "bug".

Chances are quite good that, when you run your program after typing it out, it will be error-free. Because of the SmartBASIC specific error message/prompt feature, you will have debugged your program as you went along. Most of your programs will be gold-star programs...that is, they will work on the first run. But the proper way to debug a program is a good thing to know if your interest in computers should grow, so we've inserted a brief section on efficient debugging here.

To find the bug, use three techniques:

1. Inspect the program.
2. Trace the computer's thinking
3. 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 a typing error. Maybe you:

- typed the letter O instead of zero? or
- typed zero instead of the letter O?
- typed the letter "l" instead of the number "1" or the number "1" instead of the letter "l"?
- pressed the SHIFT key when you weren't supposed to? or forgot to press it? or had the SHIFT LOCK on?
- typed an extra letter? or omitted a letter?
- typed a line you thought you hadn't? or omitted a line? or wrote over 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.

Another way to trace program execution is with TRACE and NOTRACE. You can read more about this in the Reference Section at the back of this manual.

Shorten the program

When all else fails, shorten the program.

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.

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. Use a reference guide, ask a friend, or write a tiny experimental program that contains the statement, and see what happens when you type RUN.

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 add perfected extras, so you gradually build a perfected masterpiece. This is the idea on which ADAM SmartBASIC is based. If you try to compose a long program all at once--instead of building it from perfected pieces--you could end up with nothing more than a mastermess--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; and by double-checking the foundation before you start adding the walls and the roof.

Editing

To list your entire program, just type:

LIST

Here's how to list just line 20:

LIST 20

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

LIST 20-50

or

LIST 20,50

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

LIST 20-

or

LIST 20,

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

LIST ,20

To delete line 20, just type:

20 (and press RETURN)

To delete lines 20-50, type this:

DEL 20-50

or

DEL 20,50

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

NEW

Pause

To freeze your computer screen while your program is listing, hold down the CONTROL key. While you are holding this key down, lightly tap the S key.

After you've frozen the screen, read what's on it. Then you have two choices: either abort your program with a CONTROL-C or make the computer resume printing. Here's how to make the computer resume printing (onscreen only): hold down the CONTROL key and lightly tap the S key. Right. It's the same way you stopped it!

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 GOTO 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 GOTO 20
```

Lines 20-40 form a loop. (Like most loops, the loop's bottom line says GOTO. 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; 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 ERROR IN 20
```

Then the computer stops.

So the last three lines the computer prints are:

```
ONIONS  
PEAS  
OUT OF DATA ERROR IN 20
```

Instead of saying OUT OF DATA/ERROR IN 20, 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 DONE:

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

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

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

The DONE at the end of the data is called the end mark, because it marks the end of the data. The routine that says:

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

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$: IF A$="DONE" THEN PRINT "THOSE ARE MY  
FAVORITE FOODS": RESTORE: GOTO 10
```

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  
etc.
```

The computer will print copies of its list of foods again and again, forever, unless you abort your program.

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 GOTO, so that you create a READ loop. Your program consists of two parts: the DATA, and the READ loop.

Debts

Suppose these people owe you things:

<u>Person</u>	<u>What person owes</u>
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,
HARRY,A STEAK DINNER AT MARIO'S,MOMMY,A KISS

If you decide to split a long DATA statement up into two or more smaller DATA statements, remember

to leave the comma off the ends of the DATA lines. To be safe, you can insert quotation marks, like this:

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

On ADAM, the quotation marks are unnecessary as long as your data string itself doesn't contain a comma or a colon.

In this program, the data comes in pairs: the first pair consists of BOB and \$537.29. Tell the computer to READ each pair of DATA:

```
20 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:

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

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

```
60 PRINT  
70 PRINT
```

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

```
80 GOTO 20
```

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 ERROR IN 20

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

I'VE FINISHED WRITING THE LETTERS

To do that, put a DONE at the end of the data, twice:

"HARRY", "A STEAK DINNER AT MARIO'S", "MOMMY",
"A KISS", "DONE", "DONE"

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

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

You need two DONE's 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:

<u>Person</u>	<u>Weight before</u>	<u>Weight after</u>
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
```

You can place your DATA information within quotes, if you prefer. Use of quotes on ADAM is optional and up to the user's discretion. Here is what your DATA statement would look like within quotes:

```
10 DATA "JOE",273,219,"MARY",412,371,  
"BILL",241,173  
20 DATA "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 GOTO 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
ERROR IN 20. Instead, let's make it say:

COME TO OUR DIET CLINIC!

To do that, put a DONE and two zeros at the end of
the data:

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

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

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

You need the two zeros after the DONE, 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 ERROR IN 20. 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:

<u>English</u>	<u>French</u>
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 GOTO 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	50 IF E\$<>C\$ again: THEN GOTO 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 ERROR IN 30

The computer says OUT OF DATA ERROR IN 30 because it can't find PINK in the data. Instead of saying OUT OF DATA ERROR IN 30, 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 DONE at the end of the data; and when the computer reaches the DONE, 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
30 DATA GREEN,VERT,BLUE,BLEU,BROWN,BRUN,BLACK,
NOIR,DONE,
40 READ E$,F$: IF E$="DONE" THEN PRINT "I WASN'T
TAUGHT THAT COLOR": END
50 IF E$<>C$ THEN GOTO 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 GOTO 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,DONE,
DONE
40 READ E$,F$: IF E$="DONE" THEN PRINT "I WASN'T
TAUGHT THAT COLOR": GOTO 70
50 IF E$<>C$ THEN GOTO 40
60 PRINT "IN FRENCH, IT'S ",F$
70 RESTORE
80 GOTO 10

```

To exit this, and any other program containing a loop that you must break, hit CONTROL-C.

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:

```

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
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.

Party

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

```
I SAW 2 GULLS
MEET 2 BUOYS
TRA-LA-LA!
```

```
I SAW 3 GULLS
MEET 3 BUOYS
TRA-LA-LA!
```

```
I SAW 4 GULLS
MEET 4 BUOYS
TRA-LA-LA!
```

I SAW 5 GULLS
MEET 5 BUOYS
TRA-LA-LA!

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

To do that, type these lines:

```
20 PRINT "I SAW ";X;" GULLS"  
30 PRINT "MEET ";X;" BUOYS"  
40 PRINT "TRA-LA-LA!"  
50 PRINT
```

This gives you each line of each verse plus a blank line underneath each verse.

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

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

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

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

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

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;" GULLS"
30 PRINT "MEET ";I;" BUOYS"
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". We'll follow the tradition: in the rest of this book, we'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
```

now 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.

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 three 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 we picked the letter A instead. Notice that each NEXT loops back to a specific FOR. Including the variable after the NEXT makes your programs easier to read; but SmartBASIC really is smart! It will keep track of which NEXT goes with each FOR automatically.

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 two loops are nested loops. Be careful if you use variables with your NEXT's or your loops won't be nested and you will get an error.

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 GOTO 100
50 PRINT "NO."
60 NEXT I
70 PRINT "SORRY, YOUR FIVE GUESSES ARE UP! YOU
LOSE."
80 END
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.

It's 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. Be careful when you use abnormal exits. It's very easy for both you and the computer to get confused. Usually it's better to use a counter and IF...THEN rather than abnormally exit a FOR...NEXT. (See Chapter Four.)

Short cuts

On ADAM, when you type NEXT, you don't have to type the variable even for nested loops. 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

```
FOR I=5 TO 17 STEP 3
```

```
FOR I=17 TO 5 STEP -3
```

Meaning

I will be every third number from 5 to 17. So I will be 5, then 8, then 11, then 14, then 17.

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 5
```

It says to start I at 5, and keep adding 5 until I gets past 16. So I will be 5, then 10, then 15. I won't be 20 inside the loop, since 20 is past 16. The first value of I is 5; the last value printed inside the loop is 15.

```
10 FOR I=5 TO 16 STEP 5
20 PRINT I;" IN THE LOOP"
30 NEXT I
40 PRINT I;" AFTER THE LOOP"
```

Prints out:

```
5 IN THE LOOP
10 IN THE LOOP
15 IN THE LOOP
20 AFTER THE LOOP
```

In the statement `FOR I = 5 TO 16 STEP 5`, the first value, or initial value of `I` is 5, the limit value is 16, and the step size or increment is 5. 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 20.

Random Numbers

Usually, the computer is predictable: it does exactly what you say. But sometimes you want the computer to be unpredictable. In games, for instance; 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.

This makes the computer print a list of unpredictable numbers from 1 to 6:

```
10 FOR I=1 TO 10
20 PRINT INT(1+6*RND(1))
30 NEXT
```

Unpredictable numbers are called random numbers. Line 20 makes the computer print a random number from 1 to 6. So the computer will print 1 or 2 or 3 or 4 or 5 or 6; you can't predict which of those numbers the computer will print; the computer's choice will be a surprise.

To make the computer print many such random numbers, say `GOTO`:

```
10 PRINT INT(1+5*RND(1))
20 GOTO 10
```


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 abort your program. Use Control-C to break into the loop and stop the RUN.

To get sequences of random numbers that are the same each time, use a negative number inside the RND's parentheses.

```
10 INPUT N
20 LET N=RND(-N)
30 PRINT INT(1+RND(1))
40 GOTO 30
```

This will print a series of 0s and 1s that will be the same each time the program is run. Change the sequence by INPUTting a new value for N.

We can make random work better by using the DEF FN statement. At the same time, we'll use less paper (or less screen) by making the computer print the numbers across instead of down.

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

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

```
10 DEF FN D(N)=INT(1+N*RND(1))
20 PRINT FN D(5); " ";
30 GOTO 20
```

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

```

10 DEF FN D(N)=INT(1+N*RND(1))
20 PRINT FN D(1000);" ";
30 GOTO 20

```

The computer will print something like this:

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

This program plays a guessing game:

```

5 INPUT N
6 N=RND(-N)
10 DEF FN D(N)=INT(1+N*RND(1))
20 PRINT "I'M THINKING OF A NUMBER FROM 1 TO 100."
30 C=FN D(100)
40 INPUT "WHAT DO YOU THINK MY NUMBER IS? ";G
50 IF G<C THEN PRINT "YOUR GUESS IS TOO LOW.": GOTO
40
60 IF G>C THEN PRINT "YOUR GUESS IS TOO HIGH.": GOTO
40
70 PRINT "CONGRATULATIONS! YOU FOUND MY NUMBER!"

```

Input any one, two, or three digit number when asked by the computer.

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 GOTO 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 GOTO 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 talk about your friends:

```
5 INPUT N
6 N=RND(-N)
10 DEF FN D(N)=INT(1+N*RND(1))
20 INPUT "TYPE THE NAME OF SOMEONE YOU LOVE...? ";N$
30 R=FN D(3)
40 IF R=1 THEN PRINT N$;" WEARS LOUD, STRIPED
SOCKS. ARE YOU SURE YOU WANT TO GET INVOLVED?":
GOTO 20
50 PRINT N$;" LOVES YOU, TOO": GOTO 20
```

Note that lines 5 and 6 are a kind of internal RESET in this program. This is called a "seed". This two-line "seed" is what tells ADAM which random sequence to use.

Type in any one, two, or three digit number when the question mark appears.
Line 20 makes the computer ask:

TYPE THE NAME OF SOMEONE YOU LOVE...?

Suppose the human says SUZY. The 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 WEARS LOUD, STRIPED SOCKS. ARE YOU SURE YOU WANT TO GET INVOLVED?

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 WEARS LOUD, STRIPED SOCKS etc. The chance is 2 out of 3 that the computer will be nicer, and say LOVES YOU, TOO. Here's a sample run:

```
RUN
? 123
TYPE THE NAME OF SOMEONE YOU LOVE...? SUZY
SUZY LOVES YOU, TOO
TYPE THE NAME OF SOMEONE YOU LOVE...? JOAN
JOAN WEARS LOUD, STRIPED SOCKS. ARE YOU SURE YOU
WANT TO GET INVOLVED?
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 WEARS LOUD, STRIPED SOCKS. ARE YOU
SURE YOU WANT TO GET INVOLVED?
etc.
```

This program predicts what will happen to you today:

```
5 INPUT N
6 N=RND(-N)
10 DEF FN D(N)=INT(1+N*RND(1))
20 PRINT "YOU WILL HAVE A ";
30 R=FN D(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.

Print Zones

What are zones and how do they work? The leftmost part of the screen is the first zone; to the right of that zone lies the second zone. Each print zone is 16 characters wide. This allows 2 zones per line on the screen and 5 zones per line on the printer.

A comma makes the computer jump to a new zone; here's an example:

```
10 PRINT "THIN","KING"
```

The computer will print THIN and KING on the same line; but because of the comma before KING, the computer will print KING in the second zone, like this:

```
THIN          KING
```

This program does the same thing:

```
10 PRINT "THIN",  
20 PRINT "KING"
```

Line 10 makes the computer print THIN and then jump to the next zone. Line 20 makes the computer print KING. The computer will print:

```
THIN          KING
```

This program makes the computer greet you:

```
10 PRINT "HI","HI"
```

The computer will print HI two times; each time will be in a new zone, like this:

```
HI           HI
```

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.

Here's another program:

```
10 PRINT "YOU'RE NICE", "...LIKE ME!"
```

The computer will print YOU'RE NICE, then jump to a new zone, then print ...LIKE ME! like this:

YOU'RE NICE

...LIKE ME!

LESSON 4

TACKLE THE TOUGH STUFF

Incrementing

Here's a strange program; until you stop and think about it:

```
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 one. See if you can see the logic behind it:

```
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 **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 incrementing:

```
10 C=3
20 PRINT C
30 C=C+1
40 GOTO 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 abort your program.

Here's the general procedure for making the computer count:

1. Start 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 GOTO 20
```

The semicolon makes the computer print horizontally:

3 4 5 6 7 8 9 etc.

This program makes the computer count, starting at 1:

```
10 C=1
20 PRINT C;" ";
30 C=C+1
40 GOTO 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 GOTO 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 GOTO 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
1.21	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 1:

```
5 PRINT "NUMBER", "SQUARE"  
10 C=0  
20 PRINT C, C*C  
30 C=C+.1  
40 GOTO 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 was Beethoven's first name?
- How many cups are in a quart?

To make the computer score this 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 THERE 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 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 GOTO the next question:

```
130 IF H$=A$ THEN PRINT "CORRECT": GOTO 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$: GOTO 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 ERROR IN 100
```

To give a quiz about different topic, change the data in lines 10-50.

Instead of making the computer say OUT OF DATA ERROR IN 100, let's make it say:

```
I HOPE YOU ENJOYED THE QUIZ
```

To do that, write an end mark and an end routine:

```
60 DATA "DONE", "DONE"
100 READ Q$,A$: IF Q$="DONE" THEN PRINT "I HOPE YOU
ENJOYED THE QUIZ": END
```

Let's also 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 "DONE", "DONE"
100 READ Q$,A$: IF Q$="DONE THEN PRINT "YOU ANSWERED
";C;" OF THE QUESTIONS CORRECTLY": PRINT "I HOPE YOU
ENJOYED THE QUIZ": END
110 PRINT Q$
120 INPUT "??";H$
130 IF H$=A$ THEN PRINT "CORRECT": C=C+1: GOTO 100
140 PRINT "NO, THE ANSWER IS: ";A$: GOTO 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

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 GOTO 20
```

To exit from these looping programs, hit CONTROL-C.

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 eventually 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 $S=S+\text{the number to be added}$)
5. GO back TO step 2.

Checking account

If your bank is nasty, it treats your checking account something like this: you must pay a 20 cent service charge for each good check; you must pay a \$10 penalty for each check that bounces; and the money you've deposited earns no interest.

This program makes the computer imitate such a bank.

```
10 S=0
20 PRINT "YOUR CHECKING ACCOUNT CONTAINS $ ";S
30 INPUT "DEPOSIT OR WITHDRAW? ";A$
40 IF A$="DEPOSIT" THEN GOTO 100
50 IF A$="WITHDRAW" THEN GOTO 200
60 PRINT "PLEASE SAY 'DEPOSIT' OR 'WITHDRAW'": GOTO 30
100 INPUT "HOW MUCH DO YOU WANT TO DEPOSIT? ";D
110 S=S+D
120 GOTO 20
200 INPUT "HOW MUCH DO YOU WANT TO WITHDRAW? ";W
210 W=W+.20
220 IF W<=S THEN PRINT "OKAY": S=S-W: GOTO 20
230 PRINT "THAT CHECK BOUNCED": S=S-10: GOTO 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 says 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, which 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 20 cent charge to the withdrawal amount. Line 220 compares the withdrawal amount (W) against the sum in the account (S). If $W \leq S$, 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 \$10 (the penalty fee), and goes back to line 20, for the next transaction.

Again, to exit this program, press CONTROL-C and RETURN.

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:
 $7+8+9+\dots+100$. Here's how:

Set the sum at zero:	10 S=0
Make I go from 7 to 100:	20 FOR I=7 TO 100
Add each I to sum:	30 S=S+I
	40 NEXT I
Print final sum:	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:
 $(7 \text{ squared}) + (8 \text{ squared}) + (9 \text{ squared}) + \dots + (100 \text{ squared})$. Here's how:

```
10 S=0
20 FOR I = 7 TO 100
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, 6.1, etc.
30 DATA etc.
40 DATA etc.
50 DATA 0
60 READ X: IF X=0 THEN PRINT S: END
70 S=S+X
80 GOTO 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, then 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.

Single Subscripts

Instead of being a single number, X can be a whole list of numbers, like this:

```
X=  57.2
    -8.3
    476
    2.008
    -19
    372.402
    0
    215.6
```

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 PRINT X(1),X(2),X(3)...etc.
```

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

To achieve the same results, in a single column configuration, you can say:

```
100 FOR I = 1 TO 8: PRINT X(I): NEXT
```

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. Definition: an array is a variable that has subscripts. An array may be multidimensional in that it may have 1 to 88 dimensions.

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 FOR I = 1 TO 8: READ X(I): NEXT
40 FOR I = 1 TO 8: PRINT X(I): NEXT
```

Let's make the program fancier, by adding extra lines. This line makes the computer add the eight 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 eighth 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

So, if you run this program, you'll get the initial listing, the total, and the initial listing again, but in reverse order.

Analysis

Suppose you want to analyze 50 numbers. Begin your program by saying:

```
10 DIM X(50)
```

Then type 50 numbers, as data, like this:

```
20 DATA etc.
```

```
30 DATA etc.
```

```
40 DATA etc.
```

Tell the computer to READ the data:

```
100 FOR I = 1 TO 50: READ 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 $X(1)+X(2)+X(3)+\dots+X(50)$.

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":END
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 is 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 = 2 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 = 2 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 ORDER": END
130 NEXT I
```

otherwise, print YES:

```
140 PRINT "YES, THE LIST IS IN STRICTLY INCREASING
ORDER"
```

Multiple Arrays

Suppose your program involves two lists of numbers. Suppose the first list is called A and the second list is called B. Suppose A's list contains 18 numbers and B's list contains 57 numbers. To say all that, begin your program with this statement:

```
10 DIM A(18), B(57)
```

Double Subscripts

You can make X be a table of numbers, like this:

X=	57	8.4
	-6	1000
	0	7.77

Here's how to make X be that table:
Begin by saying:

```
10 DIM X(3,2)
```

That says X will be a table, having 3 rows and 2 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(2,1)=-6
50 X(2,2)=1000
60 X(3,1)=0
70 X(3,2)=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 FOR I = 1 TO 3: FOR J = 1 TO 2: 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(10,2)
20 FOR I = 1 TO 10
30 FOR J = 1 TO 2
40 X(I,J)=I*J
50 NEXT J
60 NEXT I
70 FOR I = 1 TO 10: FOR J = 1 TO 2: PRINT
X(I,J),:NEXT: PRINT: NEXT
```

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
-8	402
5106	-.2
36.0	.04
777	666
1.99	2.99
50	40
12	21
0	1000

The table has 9 rows and 2 columns; so begin your program by saying:

```
10 DIM X(9,2)
```

Each row of the table becomes a row of the data:

```
11 DATA 32.7, 19.4
12 DATA -8, 402
13 DATA 5106, -.2
14 DATA 36.0, .04
15 DATA 777, 666
16 DATA 1.99, 2.99
17 DATA 50, 40
18 DATA 12, 21
19 DATA 0, 1000
```

Make the computer read the data:

```
20 FOR I = 1 TO 9: FOR J = 1 TO 2: READ X(I,J):
NEXT: NEXT
```

To make the computer print the table, say this:

```
30 FOR I = 1 TO 9: FOR J = 1 TO 2: 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:

```
100 FOR I = 1 TO 9
120 FOR J = 1 TO 2
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 8158.82

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 2
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 52.1
THE SUM OF ROW 2 IS 394
THE SUM OF ROW 3 IS 5105.8
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 2
110 print the sum of column J
120 NEXT J
```

Here are the details:

```
100 FOR J = 1 TO 2
110 S=0
111 FOR I = 1 TO 9
112 S=S+X(I,J)
113 NEXT I
114 PRINT "THE SUM FOR COLUMN ";J;" IS ";S
115 NEXT J
```

The computer will print:

```
THE SUM OF COLUMN 1 IS 6007.59
THE SUM OF COLUMN 2 IS 2151.23
```

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 ADAM, X\$ can be a list of strings, or a table of strings. For example, you can make X\$ be this table:

	DOG	CAT
X\$=	WOOF	MEOW
	HOTDOG	CATSUP

by saying this:

```
10 DIM X$(3,2)
20 X$(1,1)="DOG"
30 X$(1,2)="CAT"
40 X$(2,1)="WOOF"
etc.
```

or by saying this:

```
10 DIM X$(3,3)
20 DATA DOG,CAT,WOOF,MEOW,HOTDOG,CATSUP,
30 FOR I = 1 TO 3: FOR J = 1 TO 2: READ X$(I,J):
NEXT: NEXT
```


LOW RESOLUTION GRAPHICS

UNDERSTANDING CONCEPTS

The only concept you really need to understand in this section is how graph paper works. "It doesn't work, it just sits there". . . I know, I know. But for accuracy, it might be a good idea to take out a piece of graph paper, since the screen doesn't divide itself into an actual grid pattern that you can see. However, if you're into the more ethereal and less concrete, then picture, in your mind, a piece of graph paper. On this graph paper, there are 40 columns and 40 rows, with space for four lines of text at the bottom. Think of your screen as if it was this piece of graph paper. Type GR. You will notice that the screen has gone black. Don't worry, you didn't kill it. You'll also see that the text cursor is now at the bottom of the screen, and your text has been cleared.

Drawing with ADAM

Now, since man cannot live by black-and-white alone you have 16 colors to choose from, merely by typing COLOR= at this time. The chart below will show you which color options you have, and the codes you must enter in order to use them:

<u>Code</u>	<u>Color</u>
0	black
1	magenta (purple)
2	dark blue
3	dark red
4	dark green
5	grey
6	medium green
7	light blue
8	dark yellow (orange)
9	medium red
10	grey
11	light red
12	light green
13	light yellow
14	cyan (aqua)
15	white

NOTE: The listed colors will appear somewhat different, depending upon what sort of television set your ADAM is hooked up to. Also--if you want color graphics, ADAM must be hooked up to a color television. You won't see color on a black and white set!

To get ADAM to draw for you, in straight lines, you have to input instructions on which way to draw the lines, vertically or horizontally.

Horizontal Lines

The command for a horizontal line is HLIN. You must type this in, along with a location on the screen grid (remember the graph paper?). Here's an example:

```
HLIN 5,20 AT 35
```

5,20 indicates the column in which the line is to begin and end. This one would, therefore, begin at column 5 and end at column 20. AT 35 designates the row in which the line will be drawn. The 5 and the 20 are known as first end point and second end point, respectively. The first end point may be greater than, equal to, or less than the second end point. Remember that these "points" are locations and not values.

Try running this program:

```
10 GR
20 COLOR=14
30 HLIN 10,35 AT 20
```

Line 20 sets the color to cyan (a shade of blue). Line 30 tells ADAM to draw a line in row 20 from column 10 to column 35. Another thing to remember is that the first column and the first row are not numbered 1, but 0. Don't worry, you still have 40 columns, but they're numbered 0 to 39, and you still have 40 rows, too, which are also numbered 0 to 39.

Right now you're probably mumbling to yourself about people who tell you that no math is required to program ADAM. Take heart--this isn't really math, so stop playing with your mental blocks and put them away. What you really have in front of you is a space-age version of an Etch-A-Sketch. Remember those? Red plastic with two white dials--one for vertical and one for horizontal? And you could erase the screen by shaking it up and down? Well, DON'T shake ADAM to erase the screen! This gets expensive. The analogy may be a bit simplified, but for our purposes, the idea of plotting coordinates may frighten you less if you think of your computer graphics as an electronic Etch-A-Sketch.

Vertical Lines

So. We have horizontal lines. On to vertical. It's almost the same set-up as HLIN, except the command is VLIN. The main difference between VLIN and HLIN is this:

HLIN plots column coordinates first and uses a row coordinate for an end point.

VLIN plots row coordinates first and uses a column coordinate for an end point.

Oh, it is not either confusing. Try this program for VLIN:

```
10 GR
20 COLOR=1
30 VLIN 10,21 AT 20
```

Line 20 sets the color to magenta. Line 30 tells your computer to draw a vertical magenta line from row 10 to row 21 in (AT) column 20.

Are you beginning to see the creative possibilities in this? Are you parents thinking, "Hmmm, this could be a great way to keep the kids busy!"? Are you kids thinking, "Hmmmm, this could be a great way to keep my parents busy!"?

Block Coloring

If you want to move from line drawing into filling in a whole square, or "block" on your "graph-paper" grid, you'll need the PLOT statement.

Try this one:

```
10 GR
20 COLOR=13
30 PLOT 20,2
```

What happened on your screen? Did you get a block of yellow (13) at column 20, row 2? You can create a pattern on your screen by writing up a long program that changes the colors many times in many different grid locations. It's easy! It's fun! It's computer needlepoint!

When plotting coordinates, keep in mind that coordinate position 0,0 (column 0, row 0) is at the top left corner. 39,0 (column 39, row 0) is at the top right corner. 0,39 (column 0, row 39) is at the bottom left corner. 39,39 (column 39, row 39) is at the bottom right corner of your screen.

Here's a wild program. Run this one:

```
10 GR
20 COLOR=INT(RND(1)*16)
30 X=INT(RND(1)*40)
40 Y=INT(RND(1)*40)
50 PLOT X,Y
60 GOTO 20
```

Didn't know ADAM could dance, did you?

NOTE: Variables X and Y signify columns and rows respectively.

SCRN Function

Our next function is called SCRN. This reads the color displayed at any designated position on your

graphics screen. So, if you had done some computer coloring and needed to know what color was in the block at column 5 row 9, type:

SCRN (5,9)

and your computer will tell you what color you selected for that coordinate block.

These functions are not intended for use with high-resolution graphics. For information on the use of high-resolution graphics, please consult the reference section at the back of this manual.

When you're finished experimenting with graphics, you can return to a full text display screen simply by typing:

TEXT

And that's all there is to low-resolution graphics. Wasn't as hard as you thought it would be, was it?

For an interesting example that shows how you can use the game controllers to draw pictures, see the PDL function in the Reference Section.

Congratulations, ADAM programmer!

School's out, and you've just graduated! In successfully completing the initial learning process on your ADAM Family Computer, you are now an official SmartBASIC programmer. Should you care to press on and learn a little more, turn to the Reference Section which follows. This section discusses more advanced commands which you may find will be handy to understand for future use.