

ATARI® For Kids From 8 to 80

Michael P. Zabinski and E. Michael Scheck



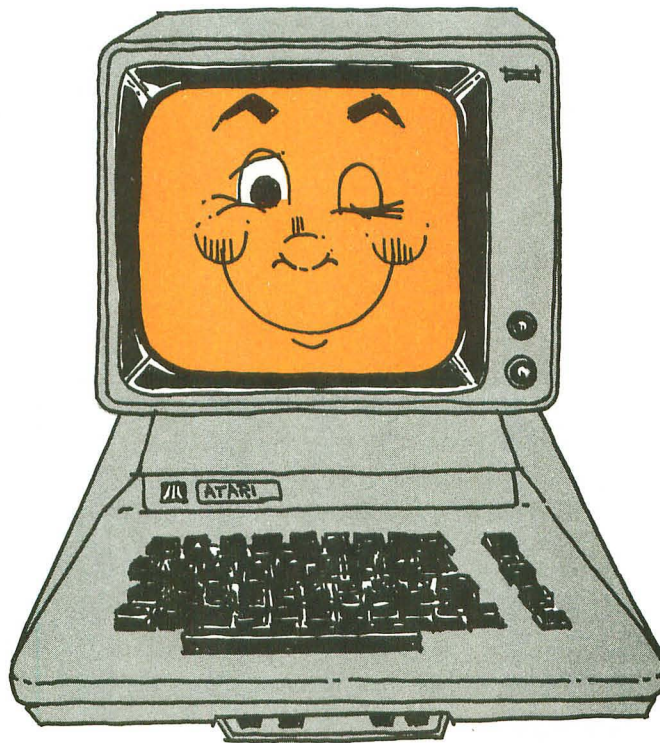
ATARI®
FOR KIDS
from
8 to 80

Howard W. Sams & Co., Inc.
4300 WEST 62ND ST. INDIANAPOLIS, INDIANA 46268 USA

Michael Zabinski, Ph.D.
Professor
Fairfield University
Fairfield, Connecticut 06430

ATARI® FOR KIDS

from
8 to 80



Written by
Michael P. Zabinski, Ph.D.
E. Michael Scheck

Designed and illustrated by Linda Yakel

Copyright © 1984 by Michael P. Zabinski, Ph.D.

FIRST EDITION
FIRST PRINTING — 1984

All rights reserved. No part of this book shall be reproduced, stored in a retrieval system, or transmitted by any means, electronic, mechanical, photocopying, recording, or otherwise, without written permission from the publisher. No patent liability is assumed with respect to the use of the information contained herein. While every precaution has been taken in the preparation of this book, the publisher assumes no responsibility for errors or omissions. Neither is any liability assumed for damages resulting from the use of the information contained herein.

This book is published by Howard W. Sams and Co., Inc. which is not affiliated with Atari, Inc. and Atari is not responsible for any inaccuracies.

International Standard Book Number: 0-672-22294-9
Library of Congress Catalog Card Number: 84-50178

Printed in the United States of America.

INTRODUCTION FROM THE AUTHORS

COMPUTERS ARE FOR KIDS is the slogan of the National Computer Camps® founded in 1977 in Orange, Connecticut. Since then kids from all over have attended the camps. They all have one thing in common: **THEY LOVE COMPUTERS**. Comments by some boys and girls are: "It's fun and smart." "... it makes me think." "I can't wait to grow up and become a computer engineer."

The ATARI Computer System, including the 400, 800, and XL series, is especially easy to learn to operate and program. The ATARI Home Computer is excellent at using color and sound. Kids of all ages enjoy making the Computer do as it is told. Yes the ATARI Home Computer is our friendly servant who always does as he is told. "Telling the Computer" is called programming. You're the boss — but you need to learn to talk its language. What language is that? Computer-eze? ATARI ...? or ... Well, the people who designed the ATARI Home Computer knew this would be a problem, so they designed it to speak a language called BASIC ... And fortunately, BASIC is all simple English words — so it's not like you have to learn a foreign language!

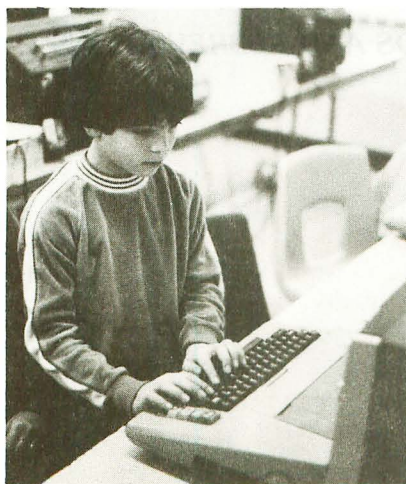
This book is written especially for youngsters who want to learn to program the ATARI Home Computer. No special background is needed. Only the most important parts of BASIC are covered and the material is presented in a light and entertaining manner. Each reader should run as many programs as possible since computer programming is best learned by doing. A variety of activities make the book stimulating and interesting.

The last chapter of the book, "Programs To Go" contains a series of programs which are educational and recreational. Several of these programs were written by Marilyn Scheck.

As you will soon see, working with the Computer means learning and having fun!

Michael P. Zabinski, Ph.D.
E. Michael Scheck

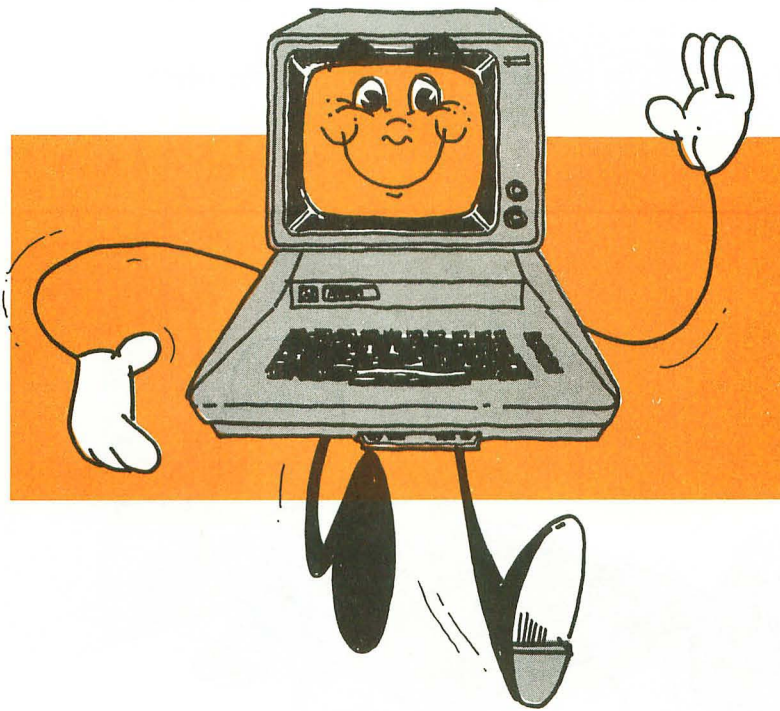
NATIONAL COMPUTER CAMPS® PHOTOS



Computer programming — an enjoyable experience.

TABLE OF CONTENTS

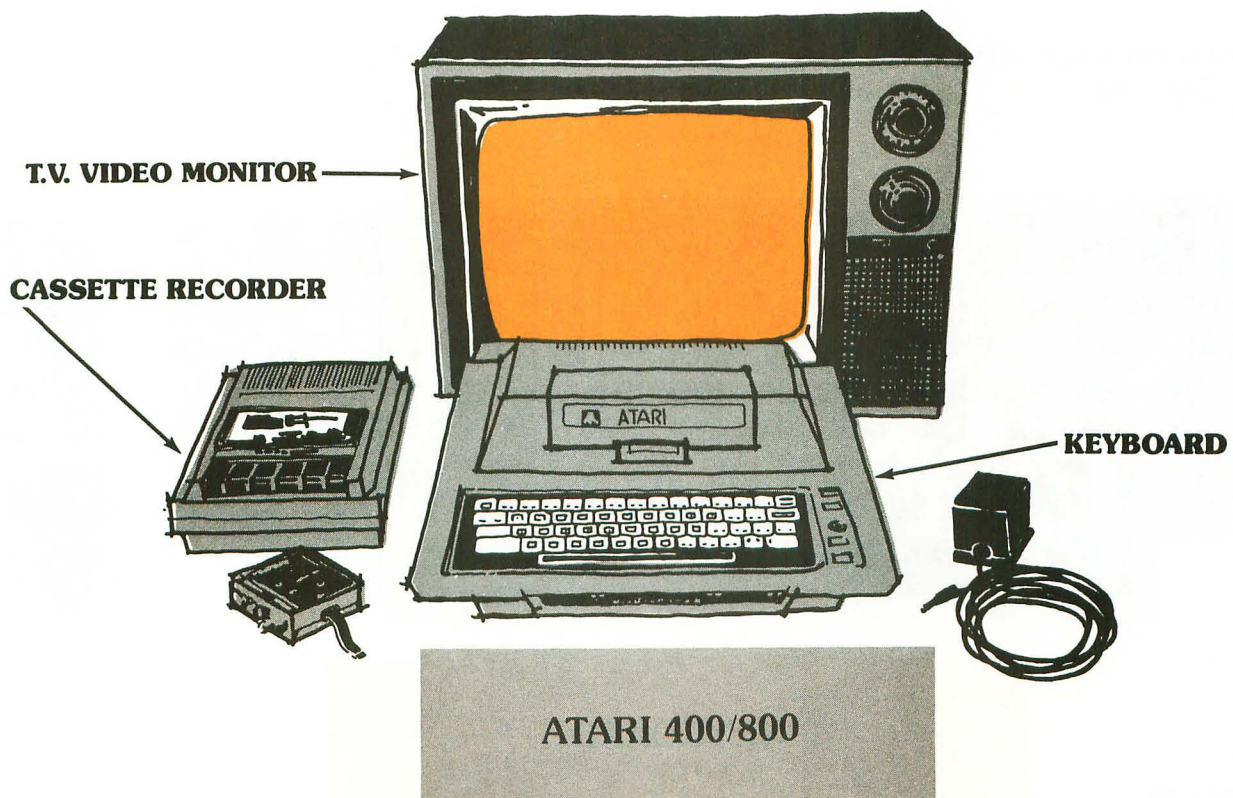
	PAGE
CHAPTER 1 TRY IT — YOU'LL LIKE IT	1
CHAPTER 2 COMMAND PERFORMANCE	7
CHAPTER 3 FIX-IT SHOP WITH COLOR & SOUND	13
CHAPTER 4 RUNNING IS GOOD FOR YOU	23
CHAPTER 5 READING IS EASY	33
CHAPTER 6 OH — TO BE A RECORDING STAR	43
CHAPTER 7 A FRIENDLY CONVERSATION	55
CHAPTER 8 IF YOU WANT TO	65
CHAPTER 9 LOOP-THE-LOOP	79
CHAPTER 10 HOW HIGH CAN YOU COUNT	93
CHAPTER 11 SURPRISE NUMBERS	107
CHAPTER 12 A RAINBOW OF LETTERS	121
CHAPTER 13 COLOR ME PURPLE	135
CHAPTER 14 PROGRAMS TO GO	153
ANSWERS	161
APPENDICES I — ATARI RESERVED WORDS AND ABBREVIATIONS	185
II — ERROR MESSAGES	186
III — GRAPHICS MODES	187
IV — COLOR CHART	188
V — SOUND CHART	189
INDEX	190



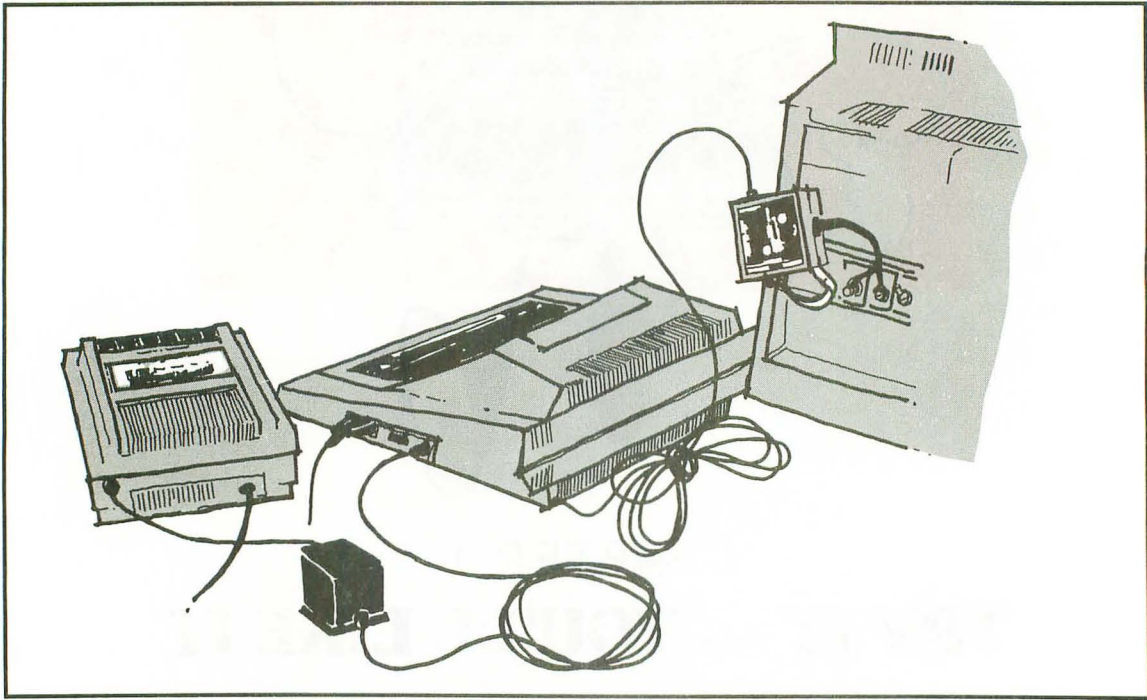
CHAPTER 1

TRY IT — YOU'LL LIKE IT

The ATARI® 400™/800™ Computer series consists of the keyboard, the television screen (or monitor), and the cassette recorder.



These parts of the Computer need to be connected properly. You may need some help.



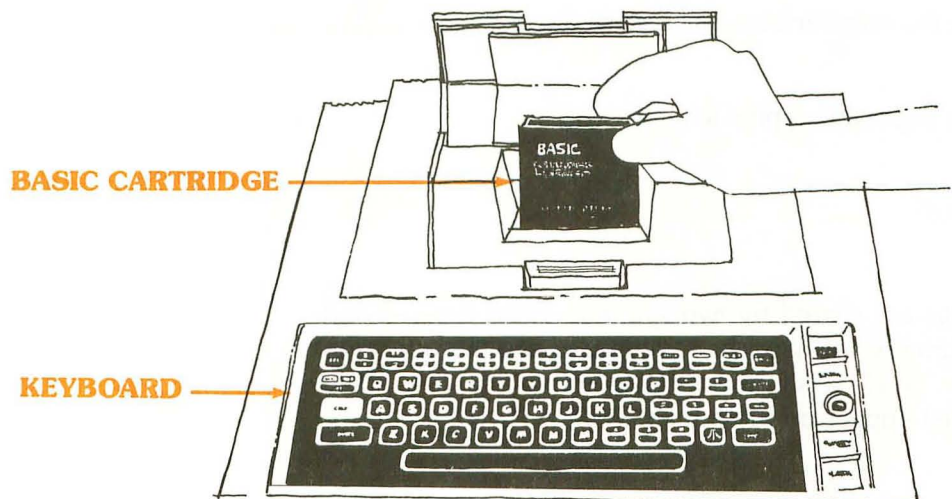
COMPUTER WITH ALL PARTS CONNECTED

The keyboard is like a typewriter. We type instructions on it; that's how we "talk" to the Computer.



KEYBOARD

Open up the top of the Computer case and insert the BASIC cartridge (in the left slot on the ATARI 800 Home Computer). Then close the top. Turn on your ATARI Home Computer and TV (or monitor). Be sure to have the cassette recorder attached first. When **READY** appears on the screen, it is ready to go.



The keyboard has keys with letters, numbers, punctuation, and many special character keys. Let's type the alphabet.

ABCDEFGHIJKLMNOPQRSTUVWXYZ

appears on the video screen.

Whatever we type always appears on the screen for us to see. To erase letters, press the **BACK S** key. Erase the alphabet. How many times do you have to press the **BACK S** key?

That's right — 26 times.

Now type your name and birth date. For example,

UNITED STATES JULY 4, 1776

appears on the screen.

To put spaces between words just press the **SPACE BAR**. To erase all of it, you press the **BACK S** key many times. Another way to erase an entire line is to press the **SHIFT** and **DELETE** keys at the same time (the **DELETE** & **BACK S** keys are the same key).

Now let us locate some other keys on the keyboard. Can you find the following keys?

LETTERS

I O X

DIGITS

1 @ 9

SPECIAL SYMBOLS

+ - *

✓ CHECKPOINT

1. Can you figure out why the number zero has a slash through it? _____
2. Are all the letters on the keyboard capital letters? _____
3. What are the special keys? _____

On the keyboard some keys are shared by two symbols. Press the **" / 2** and **SHIFT** keys at the same time, what happens? Right, you see the quotation marks appear. Press the **SHIFT** key along with the **SEMICOLON** key and watch the colon (2 dots) appear on the screen.



EXPERIMENT

The best way to learn about the keyboard is to experiment with it. You can't break it . . . so go ahead, play with it. Try out all the keys.

My favorite key is _____.

When you're done experimenting, press the **SYSTEM RESET** key. Wow, what happened?

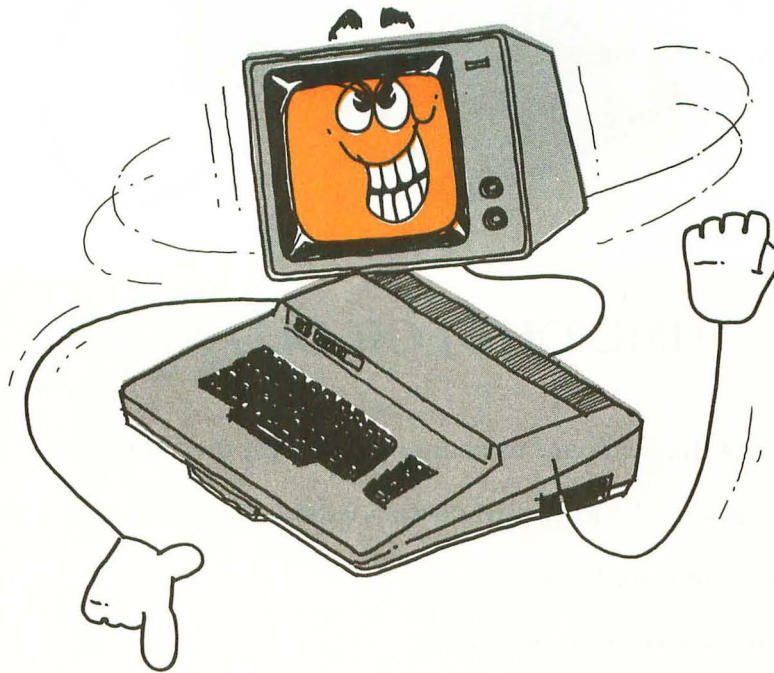
That's an easy way to erase all the clutter on the screen.

In your experimenting did you try the **CTRL** (control) key? How about the **ALT** key? If you didn't, try this:

Hold down the **CTRL** key and type at the same time. Kinda hard to read? But think of the pictures you could draw! Now push the **ALT** key once. Nothing happened, yet! Play with the keyboard again — even the **CTRL** key. Something else!

What letters are not in the alphabet?

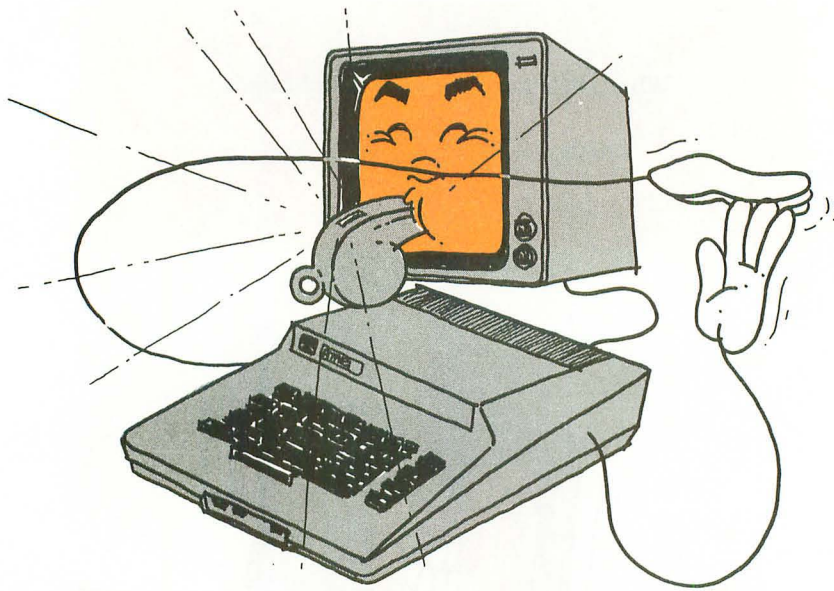
Answer: The ones in a mailbox.



CHALLENGE

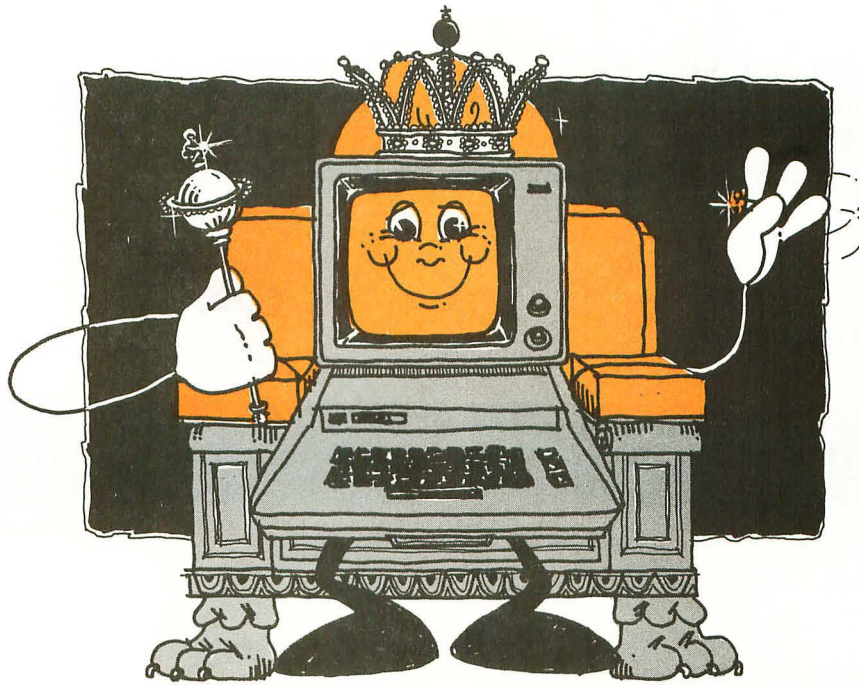
Your mission is to find out how many characters can be displayed on the screen in one line. *HINT, keep typing.*

Then find out how many lines fit on the screen. *HINT, use the **RETURN** key.*



TIME OUT FOR OLD NEWS

- You use the keyboard to communicate with the Computer.
- The keys on the keyboard contain letters, digits and special symbols.
- 24 lines of print fit on the screen.
- 38 characters fit on one line.
- The **(SHIFT) (CLEAR)** keys erase the screen.
- The **(BACK S)** key erases one character at a time.
- The **(SHIFT) (DELETE)** keys erase an entire line.



CHAPTER 2

COMMAND PERFORMANCE

The Computer is your friendly helper. It always does exactly what you tell it to do. It can do calculations and display messages for you. The keyboard is used to “talk” to the Computer. We type instructions and send them to the Computer.

When **READY** shows on the screen, the Computer is waiting for your command. Is it really? Let’s have the Computer add 1234 plus 5678.

```
PRINT 1234 + 5678
6912
```

Type this and press the **(RETURN)** key.
The Computer displays the answer.

Hey, that was fast!

SLOW MOTION INSTANT REPLAY

Type **(P)(R)(I)(N)(T)** and press the **(SPACEBAR)**, then type **(1)(2)(3)(4)** and press the **(SPACEBAR)**, then press the **(+)** key, press the **(SPACEBAR)** and type **(5)(6)(7)(8)**. So far the answer does not show. The instruction needs to be sent to the Computer. Press the **(RETURN)** key. Now the Computer quickly displays the answer.

BRAIN FOOD

Always press the **(RETURN)** key to send your instruction to the Computer.

The **PRINT** statement is our first BASIC instruction. BASIC stands for **B**eginner's **A**ll-purpose **S**ymbolic **I**nstruction **C**ode (everyone starts out as a beginner). The ATARI Home Computer understands the BASIC language and we will learn many more instructions. Appendix I lists the BASIC instructions.

YOUR TURN

Try the following arithmetic examples:



YOU TYPE THE INSTRUCTION

PRINT 12+3

*Did you remember to press **RETURN**?*

PRINT 12-3

PRINT 12*3

PRINT 12/3

THE COMPUTER RESPONDS

Copy it from the screen.

THINK

- Look at the above examples and then mark down the symbol for each of the following arithmetic operations.

OPERATION

Addition

Subtraction

Multiplication

Division

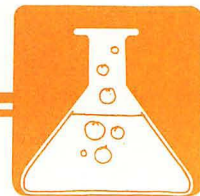
SYMBOL

Typing messages is a lot of fun. Try this one.

```
PRINT "I AM HAPPY"
I AM HAPPY
```

and press **(RETURN)**.
The message appears.

The message I AM HAPPY appears on the screen. The Computer displays the message exactly the way you type it. You need quotation marks around the message.



EXPERIMENT

Enter the following instructions exactly the way they are shown and try to predict how the Computer will respond.

INSTRUCTION

(what you type in)

DISPLAY

(what appears on the screen)

1. PRINT "MY COMPUTER LIKES ME" _____
2. PRINT "TWELVE DOZEN IS"; 12*12 _____
3. PRINT " " _____

What can you conclude from each of the above examples?

1. _____
2. _____
3. _____

BRAIN FOOD

Messages in PRINT statements must have quotation marks around them.

SHORTCUT

Our Computer knows that we all like to save time. So instead of the instruction **PRINT 3+4** it accepts **PR. 3+4**, or **? 3+4**. The **PR.** and the question mark are abbreviations for the **PRINT**. All the available abbreviations are listed in Appendix I. Try to anticipate the Computer's response for each of the following instructions.

INSTRUCTION

(what you type in)

PRINT 365*100

PR. 365*100

? 365*100

DISPLAY

(what the Computer screen shows)

CHALLENGE

What is the Computer's response to the instruction

PRINT "5 + 3"

Can you explain it?

BRAIN FOOD

PRINT displays on the screen; LPRINT displays on the printer.

FUN TIME

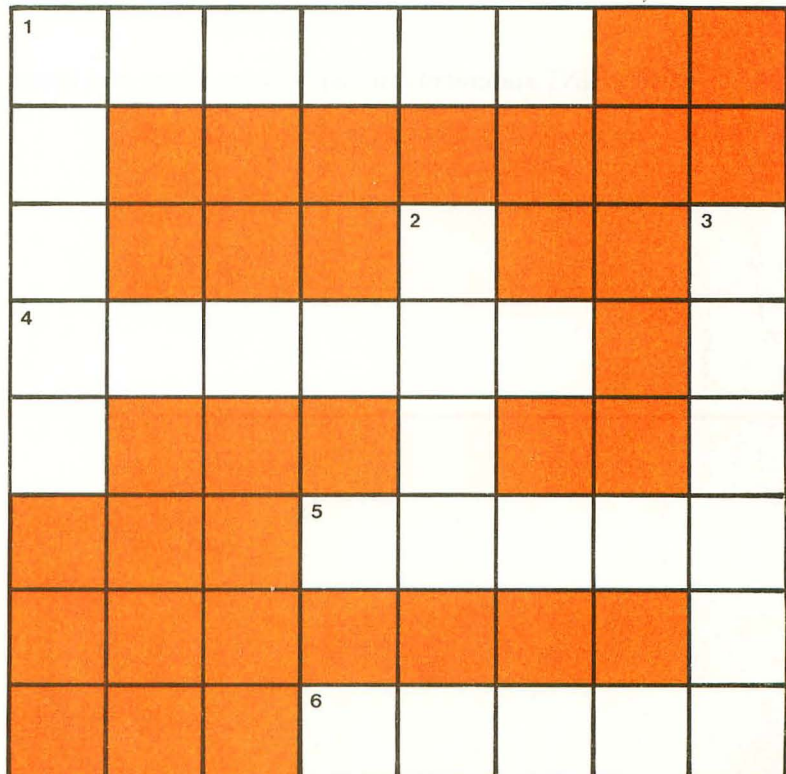
Fill in this crossword puzzle.

ACROSS

1. Press this key to send instructions to the Computer.
4. Key to erase 1 line.
5. A BASIC statement.
6. Key to erase 1 character.

DOWN

1. When the Computer is waiting for your command.
2. Symbol for multiplication.
3. Used to enclose a message.



EXERCISES

1. What display will the following instruction produce?

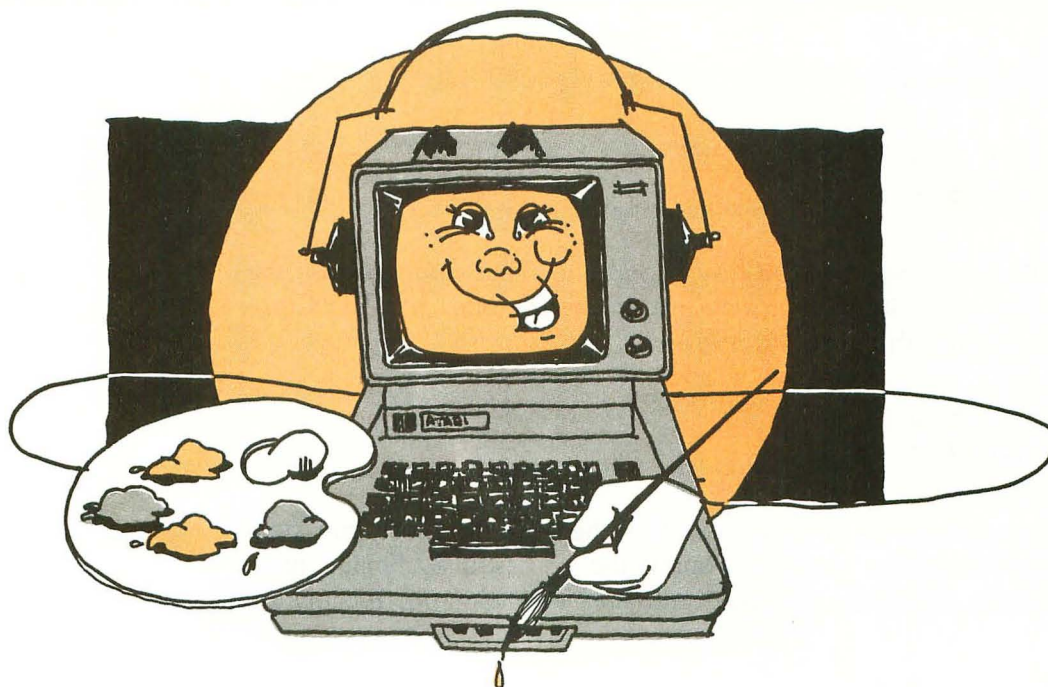
```
PRINT "AROUND THE WORLD IN ";40+40;" DAYS"
```

2. Write a BASIC instruction to figure out the number of minutes in one week.

3. Write an instruction to calculate the height of a six foot person in centimeters. There are 12 inches in one foot and 2.54 centimeters in one inch. Make the display come out in this form:

```
SIX FEET EQUAL _____ CENTIMETERS.
```

4. Use two PRINT statements to display your name and home address.



CHAPTER 3

FIX-IT SHOP WITH COLOR & SOUND

We humans sometimes make mistakes. But the Computer helps us catch them by displaying error messages.

Try this; I know it's wrong:

```
PRINT 3+4  
ERROR - PRINT            +4
```

What you type.

How the Computer responds.

This is an **error message**. Do you notice the 3 is in a white block? That is how the Computer shows you where the error is.

Now, how do you correct it?

You can type the entire line over again, or you can use the CTRL key. Let's try it!

When you hold down the CTRL (control) key the other keys do special things.

Do you see the 4 arrows over by the RETURN key? Try them out. Try to move the **cursor** (that's the white block) onto the E of ERROR —.

Press the CTRL key together with the ↑ key.

Now that you're there you have two jobs: erase the ERROR —, and retype both the M and the 3

To eliminate the ERROR — all you have to do is delete it. Hold down the CTRL key and press the DELETE BACK S key 8 times. Your Com-

puter screen should look like this:

```
PRINT 3+4
PRINT 3+4
```

What you typed.

What remains of the error message.

Try to move the cursor to the M in the second line. That's right — use the **CTRL** key with the **→** key. When you are on the M, type **N**. The Computer replaces the M with the N.

```
PRINT 3+4
PRINT 3+4
```

PRINT needs to be fixed.

Now it reads PRINT.

Move the cursor again. This time position it on the **3**. You must type the 3 over again, the Computer does not know the **3** is a 3. Finally press **RETURN** to tell the Computer that you have finished making corrections on that line.

So here's what we have done:

```
PRINT 3+4
PRINT 3+4
7
```

What you typed in.

The corrected line.

The Computer's response.

Let's correct another line.

```
PPRIIIIT 6*4+8
ERROR - PPRIIIIT 6*4+8
```

What you type.

How the Computer responds.

First let's get rid of the **ERROR** — Move the cursor to the beginning of the error line (that's right **CTRL** and the arrows).

To delete the **ERROR** —, hold down the **CTRL** and press the **DELETE BACK S** key 8 times.

Presto! The error message is gone, **but not the error**. Now to fix our line. Move the cursor again, do you know where? That's right, move it onto the first P of the second line. This time we want to delete the P so press the **CTRL** and **DELETE BACK S** keys together, once.

```
PPRIIIIT 6*4+8
PRIIIIT 6*4+8
```

What you typed.

What remains of the error message.

What's that? Only one character is gone? What do you know! Now get rid of the extra I's. Use the **CTRL** and **→** keys together to move the cursor. Now press the **CTRL** and **DELETE BACK S** keys several times. Here's what we have.

```
PPRIIIIT 6*4+8
PRIT 6*4+8
```

What you typed.

What remains of the error message.

What's the matter? Oh — **PRIT** isn't quite right, is it? How do you **INSERT** the N? Move the cursor over the T. Then find the **INSERT/>>** key and press it together with the **CTRL** key. Now type **N** in the space, and of course press **RETURN**. Don't forget to retype the **6**.

```
FFRIIIIT 6*4+8
PRINT 6*4+8
32
```

*What you typed in.
The corrected line.
The Computer's response.*

Go ahead, move the cursor around the screen and play around some more. Inserting and deleting letters is called *editing*.

BRAIN FOOD

CTRL will insert 1 space with **INSERT/>>**, or delete 1 character with **DELETE BACK S**.

SHIFT will insert 1 line with **INSERT/>>**, or delete 1 line with **DELETE BACK S**.

By the way — you'll find a list of error messages in Appendix II.

UPPER AND LOWER CASE

While you were trying out keys, did you find that the **CAPS LOWR** key will give you lower case letters after you press it?

Press the **CAPS LOWR** key.

The Computer is now like a typewriter, lower case letters unless you hold down the **SHIFT** key.

Try this:

```
PRINT "Atari is great"
Atari is great
```

*Use the **SHIFT** key to type the
P, **R**, **I**, **N**, **T**, **"**, and **A**.*

How do you get back to all capitals? Press the **SHIFT** and **CAPS LOWR** keys together one more time.

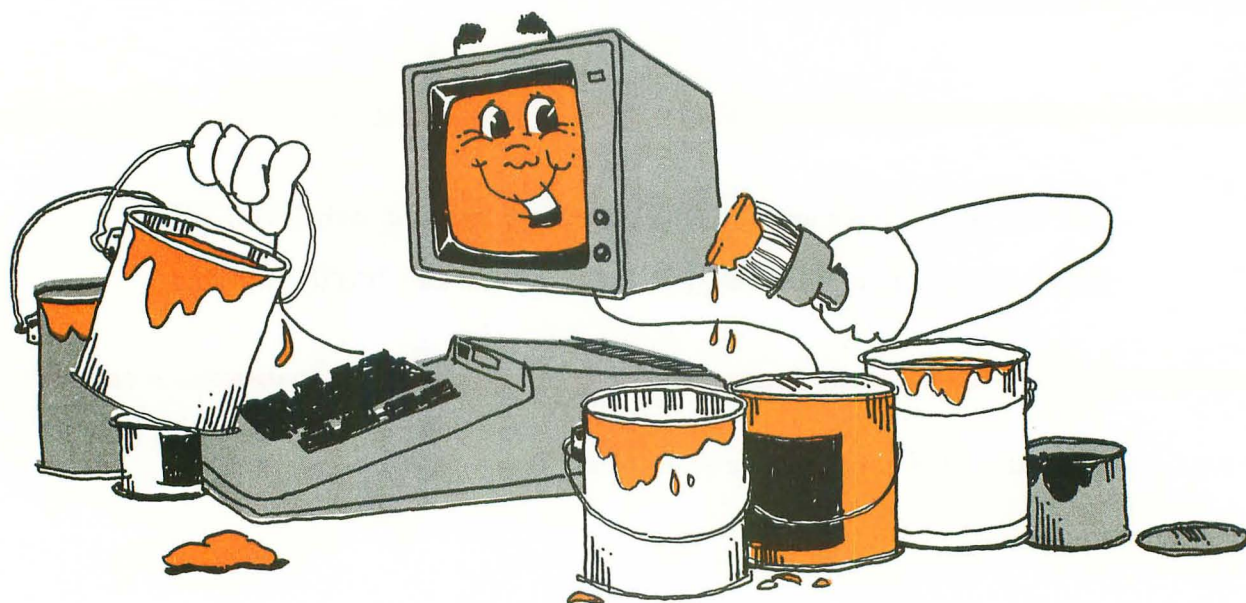
Try this:

Hold down **CTRL** and press the **CAPS LOWR** key (remember those special characters?). Now you can type the graphics characters.

Return to all caps by using the **SHIFT** and **CAPS LOWR** keys.

BRAIN FOOD

When you are stuck and can't seem to get the Computer to work, press the **SYSTEM RESET** key. It will get everything back to the way it was when the Computer was first turned on.



COLOR

Now let's do some painting.

In this chapter we will introduce the color capabilities of the ATARI Home Computer. In a later chapter there will be more on color.

Type:

```
SETCOLOR 2,4,10
```

The 4 and 10 combination colors the screen.

Look at that — Pink!

Change the middle number to anything from 0 to 15 (I'll wait). A nice rainbow of colors isn't it? Try changing the last number, any even number from 0 to 14.

```
SETCOLOR 2,4,6
```

You can choose 8 shades of each of the 16 colors. Appendix IV lists the 16

colors and their numbers.

To color the border rather than the background, change the first number of the **SETCOLOR** command:

```
SETCOLOR 4, 4, 10
```

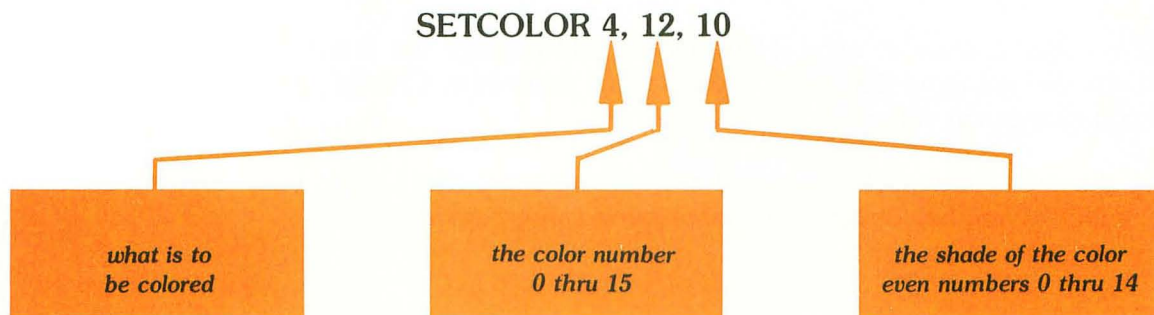
You can still use all the color and shade combinations as before.

With **2 SETCOLOR** instructions both the border and the background can be colored:

```
SETCOLOR 2, 4, 10  
SETCOLOR 4, 12, 10
```

Very Preppie!

Try it — you'll like it.



YOUR TURN

After typing in the **SETCOLOR** instruction describe the color of the screen's background or border.



WHAT YOU TYPE IN

THE COMPUTER RESPONDS

Copy it from the screen.

```
SETCOLOR 2, 5, 2
```

```
SETCOLOR 2, 4, 12
```

```
SETCOLOR 4, 12, 4
```

```
SETCOLOR 4, 14, 6
```


SOUND

Now an introduction to sound and music.

Type in SOUND 0,121,10,8

```
SOUND 0, 121, 10, 8
```

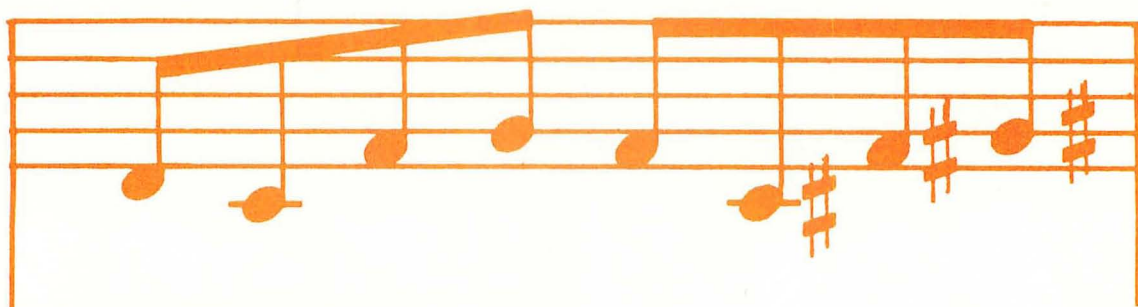
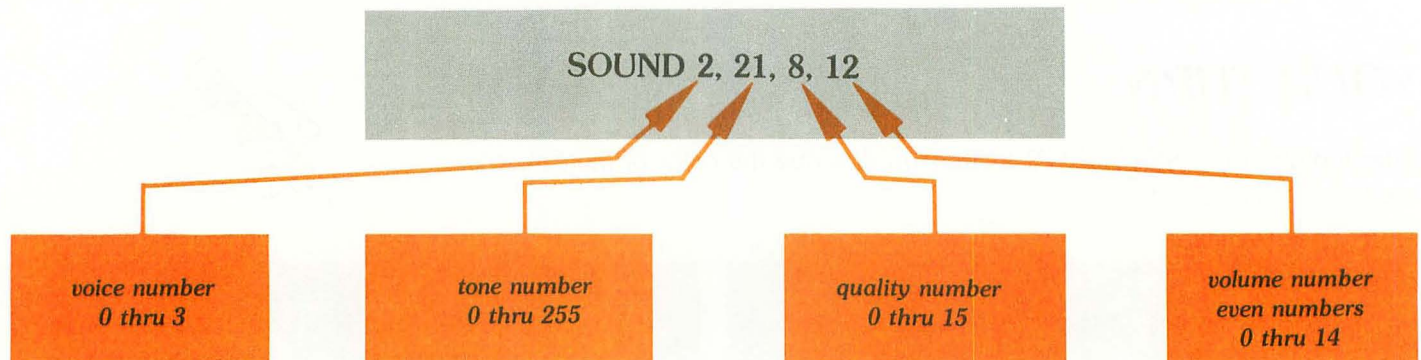
Do you recognize middle C?

If you get tired of the noise type **END** and press **RETURN**. Change the second number — anything from 0 to 255. Can you play a tune? Appendix V is a list of the sound numbers and tones. How about a chord?

```
SO. 0, 121, 10, 10
SO. 1, 91, 10, 8
SO. 2, 72, 10, 8
SO. 3, 60, 10, 10
```

You noticed! — four sounds at once. How about that? Change the first number and you can get more than one “voice” at the same time. Change the last and you change the volume.

Try changing the third number. Some weird sounds come out don't they (don't forget **END**). With the correct combinations you can get good sound effects — just like the arcades.



YOUR TURN

After typing in the **SOUND** instruction describe the tone (high/low), volume (loud/soft), quality (pure tone/harsh/buzzing/etc.).



WHAT YOU TYPE IN

SOUND 0,200,10,4

SOUND 0,100,8,10

SOUND 1,15,4,6

SOUND 1,1,1,10

SOUND 0,0,0,0

SOUND 1,151,6,6

THE COMPUTER RESPONDS

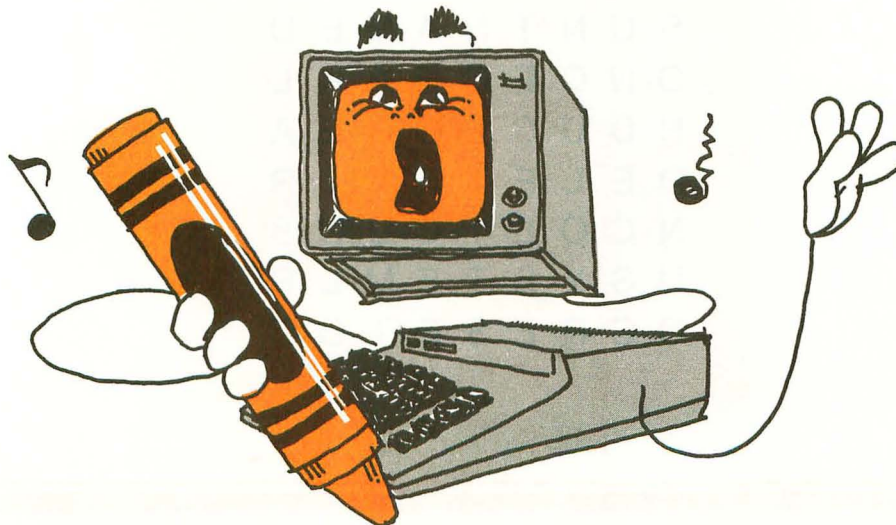
Would you like to combine sound and color? Type:

SO. 1,50,10,8 : SE. 2,6,4

*but don't press **RETURN** yet!*

Let's look at what's there — 2 instructions on one line. All you need is the colon (2 dots) between the instructions. When you press **RETURN** both instructions are executed. More than one instruction on a single line is called **chaining**.

Again try it, you may not like it, but then again you may.

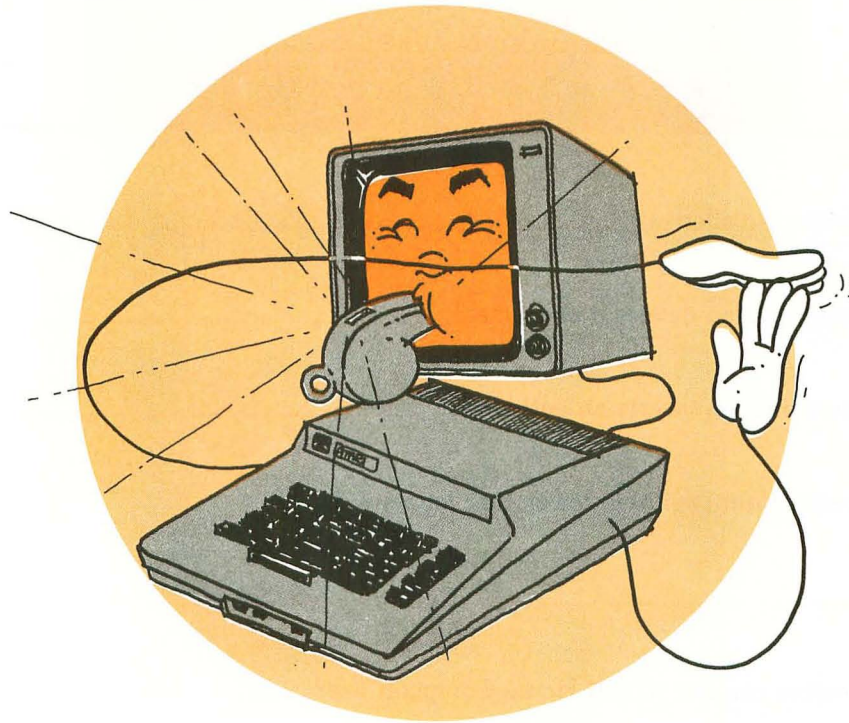


FUN TIME

Find the answers to these questions in the word search.

1. Instruction to turn on the "voices". _____
2. The key used with the arrows to move the cursor. _____
3. Instruction to change colors. _____
4. The key to erase a letter. _____
5. The key to clear the screen. _____
6. Used when all else fails (2 words). _____

D	E	L	A	S	C	M	Q	R
N	S	O	B	R	L	E	I	O
S	O	E	A	R	E	A	D	S
S	U	N	T	N	A	M	E	D
O	N	C	E	C	R	A	K	L
N	D	U	C	L	O	R	E	A
D	E	L	E	T	E	L	T	R
N	C	O	L	T	C	U	O	S
U	S	Y	S	T	E	M	L	R
O	C	R	E	S	E	T	O	C



TIME OUT FOR OLD NEWS

- The **(CTRL)** key is used to edit the screen.
- To insert 1 character press the **(CTRL)** and **(INSERT/>)** keys together.
- To delete 1 character press the **(CTRL)** and **(DELETE BACK S)** keys together.
- To delete an entire line press the **(SHIFT)** and **(DELETE BACK S)** keys together.
- The **(CAPS LOWR)** key is used to change to lower case letters.
- **(SHIFT)** and **(CAPS LOWR)** are used together to change to upper case letters.
- **(CTRL)** and **(CAPS LOWR)** are used together to change to graphics characters.
- SETCOLOR (abbreviated SE.) changes the color of the screen.
- SOUND (abbreviated SO.) activates the sound registers.
- Combining more than one instruction on one line is called chaining. Use colons between each instruction.

EXERCISES

1. Correcting errors is called _____
2. Which key combination deletes a character? _____
3. Which key combination inserts an entire line? _____
4. How do you shift from capital to lower case letters & back?

5. How many colors can the ATARI Home Computer use? _____
6. Which 2 instructions give you sound and color? _____
7. Give examples of your favorite color and sound instruction.



CHAPTER 4

RUNNING IS GOOD FOR YOU

The area of a rectangle equals its length times its width.

$$\text{AREA} = \text{LENGTH} \times \text{WIDTH}$$

If your bedroom is 12 feet long and 10 feet wide then the area is $12 \times 10 = 120$ square feet. The lengths and widths vary from room to room in your house and so do the areas.

Get it? The length, width, and area may vary. They are **variables**. We will call the variable length *L*, the variable width *W*, and the variable area *A*. Can you tell how we picked the names *L*, *W*, and *A* for these variables?

Now back to the bedroom. The length is 12 and the width is 10.

```
LET L = 12
```

L is the length. We've given it the value 12.

```
LET W = 10
```

W is the width. It is 10 feet.

```
LET A = L*W
```

The area is calculated. It equals length times width.

```
PRINT A  
120
```

Display the area.

The area equals 120 square feet.

A variable is a name that represents information stored in the Computer. We use the **LET** statement to give a variable a value.

Now for the living room:


```
LET L = 20
LET W = 15
LET A = L*W
PRINT A
300
```

*Length L is now 20.
Width W is now 15.*

The area is now 300 square feet.

The value of a variable remains in memory until it is changed.

```
PRINT L
20
LET L = 15
PRINT L
15
```

*L is still 20 feet.
Change L.*

L is now 15; it is no longer 20.

A RULE TO REMEMBER

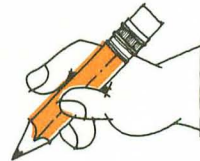
The name of a variable may be from one to 120 characters long.

The first character must be a letter. It can be followed by other letters, or digits, or a combination of letters and digits.

Proper variable names:

A AB AB EZ ROWS PAUSE1

YOUR TURN



Can you find the 3 incorrect names? Circle them.

MZ BP L+ C 4K T5 55

If you circled L+, 4K, and 55 you are correct. L+ uses a special symbol (the +), and 4K & 55 both start with a digit.

PROGRAMS

Let's turn that living room problem into a program.

A computer **program** is a set of instructions. The instructions are given to the Computer in order. It is like a boss giving a worker detailed instructions.

Here are the instructions — the program you type in.

```

NEW
10 LET L = 20
20 LET W = 15
30 LET A = L*W
40 PRINT "AREA = ";A
50 END

```

Type **NEW** and press the **RETURN** key.
 At the end of each line press **RETURN**.
W equals 15.
A is calculated.
This instruction will display the answer.

Notice that even though we have finished typing in all the instructions, no answers appear. More about that later.

We first type in **NEW**. This command erases everything in the Computer's working memory. Always type in **NEW** before you enter a program.

Each line of the program has a number. The **line numbers** tell the Computer the sequence of instructions. The lowest number is first. We skip line numbers so that we can place other instructions in between if necessary.

What is the spacing between line numbers in the above program? _____

10 is correct; the line numbers 10, 20, 30, 40, & 50 are 10 digits apart.

BRAIN FOOD

The **LET** statements give values to the variables.

NEW
RUN

Moving right along we find the last instruction to be the **END** statement. Guess what that says to the Computer. Right, it means this is the end of the program.

Your worker (the Computer) now has all of its instructions and is waiting for the boss' command . . . **RUN**.

```

RUN
AREA = 300

```

You type in **RUN** and press **RETURN**.
 Line 40 of the program displays the answer.

The **RUN** command tells the Computer to run the program.



CHECKPOINT

1. What is the first instruction of our program? _____
2. When this program is run, which line produces the display on the screen? _____
3. Now that the program is in memory, what would happen if we type **NEW**, press **RETURN**, and then type **RUN** and press **RETURN**? Try it, you may not like it!

A DISCOVERY

I am 4 feet 5 inches tall. How many centimeters is that? There are approximately 2.5 centimeters in one inch.

```
NEW
10 LET F = 4
20 LET I = 5
30 LET T = F*12+I
40 PRINT T
50 C = 2.5*T
60 PRINT C
70 PRINT F, I
80 END
```

Variable F for feet.

Variable I for inches.

Variable T for total inches.

Display the height in inches.

Compute centimeters.

Display centimeters.

The Computer remembers my height in feet and inches.

```
RUN
53
132.5
4      5
```

Ask for a run.

4 feet 5 inches equals 53 inches.

My height is 132.5 centimeters.

Or 4 feet 5 inches.

In the above example two new things crept in:

- The LET is not required.

`C = 2.5*T`

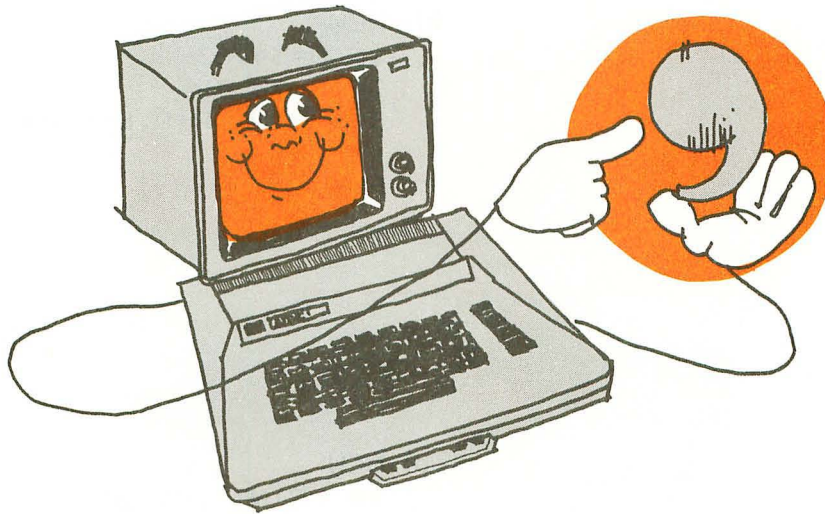
is a good as

`LET C = 2.5*T`

- We displayed the values of F and I with one PRINT statement.

```
PRINT F, I
```

The comma separates the variables F and I.



EXPERIMENT



The difference between a semicolon “;” and a comma “,” is only a dot. To the Computer this dot makes a big difference.

Type in `PRINT 1;2;3;4;5;6` and press **RETURN**.

Type in `PRINT 1,2,3,4,5,6` and press **RETURN**.

It is all in the dot! The semicolons display the numbers close together (no spaces at all). The commas display the numbers with space between them. There are four **zones** across the screen. Each of the four zones has 10 print positions.

Type in `PRINT 1,,2,,3,,4,,5,,6` and press **RETURN**.

Tricky, huh? The extra comma skips one zone between each number.

THINK

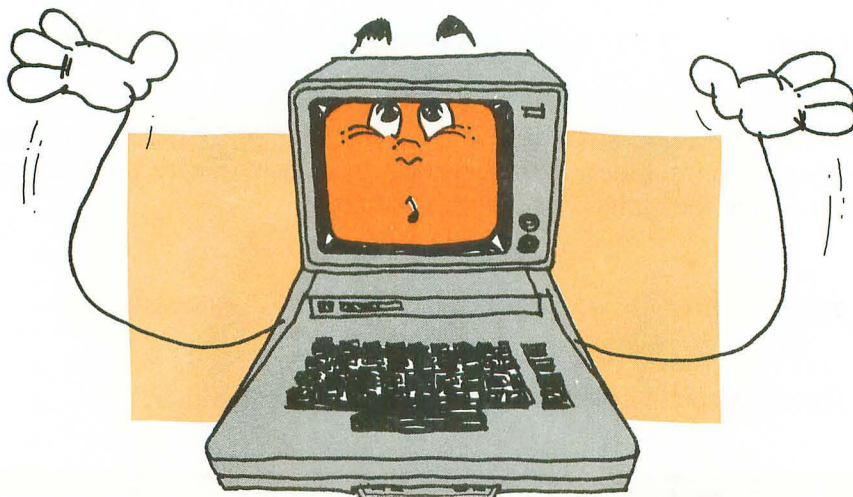
Pretend you are the Computer and run each of the following programs. (No fair cheating — try it on your own before using the Computer). Compare your answers with the Computer's responses. Watch out for spacing.

a) NEW
10 PRINT "I AM GREAT"
20 END
RUN

b) NEW
10 Q=34.53
20 R=98.32
30 PRINT R+Q,R-Q,R*Q,R/Q
40 END
RUN

c) NEW
10 A=1234
20 B=5678
30 PRINT B;"+";A,"EQUALS",A+B
40 END
RUN

d) NEW
10 PRINT 20,20*20
5 PRINT 10,10*10
15 PRINT 30,30*30
RUN



DISCOVERY

The last program, program (d), helps us discover that:

1. A spacing of 5 between line numbers is as good as a spacing of 10.
2. Each PRINT statement prints on the screen on a separate line.
3. The END statement is not necessary. You may skip it.
4. Programs are RUN in the sequence of line numbers. The line with the lowest line number first, and the highest line number last.

AHA, lot's to learn from such a short program, but there is more to come!

EXPERIMENT



The I AM GREAT program, Program (a) above, displays the message I AM GREAT. How can we change the display to I AM THE GREATEST?

Right, just edit the **PRINT** line. Using the **CTRL**/arrow combo, position the cursor between the words **AM** and **GREAT**. Insert 4 spaces — that's right use the **CTRL** and **INSERT** keys.

```
10 PRINT "I AM      GREAT"
20 END
```

Type **T H E** into the spaces. Now move the cursor to the end quote and type in the **E S T**, and press **RETURN**.

```
10 PRINT "I AM THE GREATEST"
20 END
RUN
I AM THE GREATEST
```

Now the program displays the new message.

YOUR TURN



Try this on the Computer (no line numbers this time).

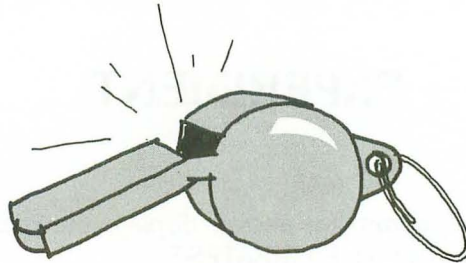
Give the variable **A** the value **1**. _____

Give the variable **B** the value **2**. _____

Write a statement to display the value of **A**. Then write a second statement to display the value of **B**.

Display the value of variable **E**.

We have not given **E** a value. That is why it is zero. Display the values of **A**, **B**, and **E** on one line.



TIME OUT FOR OLD NEWS

- A computer program consists of a set of instructions. Each instruction has a line number. The instruction with the lowest line number is run first.
- To RUN a program type in **R U N** and press **RETURN**.
- The END statement is optional.
Semicolons in a PRINT produce a compact display.
Commas in a PRINT produce a display in the 4 zones across the screen. Each zone consists of 10 spaces.
- The NEW command erases working memory. Always type **N E W** before typing in a program.

EXERCISES

1. List the two BASIC instructions you have learned so far.

2. What will be the Computer's response to this instruction?

```
PRINT "6+10 = "; 6+10
```

3. What will be the Computer's response to this instruction?

```
A+3=9
```

Can you explain the reason for the response?

4. Predict the Computer's response to the following:

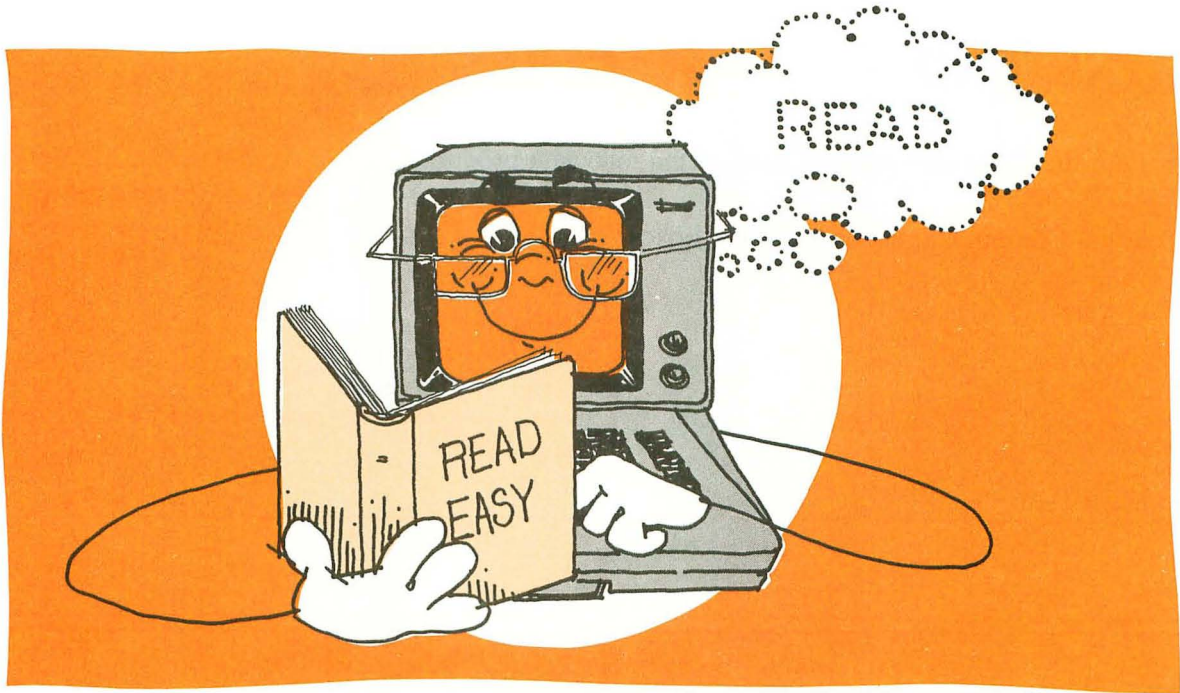
```
A = 5  
B = 6  
PRINT B+A; B-A, A+B
```

5. Predict the Computer's display for this program:

```
NEW  
10 A= 50  
5 PRINT "HELLO"  
12 B= 46  
20 PRINT A+B  
30 PRINT A*B  
25 PRINT A-B
```

6. Write a program to display the weight of a dozen eggs (1 egg = 1.4 oz.).

7. Edit the program in exercise 6 to use 50 eggs.



CHAPTER 5

READING IS EASY

Remember our living room program?

```
NEW
10 LET L = 20
20 LET W = 15
30 LET A = L*W
40 PRINT "AREA = ";A
50 END
```

*Length is 20.
Width is 15.
Area is calculated as L times W.
Display the area.*

Another way of entering the length and the width into the Computer is with a **READ** and **DATA** combination. Here are the instructions.

```
NEW
10 READ L,W
20 A = L*W
30 PRINT "AREA = ";A
40 DATA 20,15
50 END
```

Notice the comma.

The message in quotes helps us read the answer.

The **READ** statement tells the Computer to get values for variables A and B in the **DATA** statement. Here $L=20$ and $W=15$.

READ-DATA

LET'S PRETEND

Pretend you are the Computer and run each of the following programs.

Write each of the Computer's responses in the boxes.

```
NEW
10 READ A,Q
20 DATA 6,8
30 PRINT A+Q;" ";A-Q,A*Q;" ";A/Q
40 END
RUN
```

```
NEW
100 READ M,R
200 DATA 6,8,12,14
300 PRINT "2*M+3*R=";2*M+3*R
400 END
RUN
```


MORE NEWS

Numerical variables have a number as their value. For example,

`N = 5`

Numerical variable N equals 5.

String variables have a string of characters as their value. The characters can be any letters, numbers, or symbols from the keyboard. For example,

`N$ = "R2-D2"`

String variable N\$ equals the string of 5 characters R2-D2.

So what's new?

1. String variables have a \$ symbol at the end of their names.

Examples of string variable names are:

A\$

AB\$

MZ\$

R2\$

2. What the string variable is equal to, must be in quotation marks:

A\$="MARION"

R2\$="GIRL"

YOUR TURN



Underline the string variables and circle the numerical variables.

A\$ A C5\$ N\$ BS A6 A6\$

If you underlined all the variables with the dollar sign then you did great! The string variables are A\$, C5\$, N\$, and A6\$. A, BS, and A6 are numeric variables and should have been circled.

My name is ATARI 400 Home Computer (or ATARI 800 Home Computer). To enter my name into the Computer with string variable N\$, type:

```
NEW
5 DIM N$(9)

10 LET N$="ATARI 400"
20 PRINT N$
RUN
ATARI 400
```

DIM tells the Computer how long the string N\$ will be in this program.

*The quotation marks are required.
N\$ is a character string.*

To dimension a string use **DIM** followed by the string variable name and a number in the parentheses. This number is the number of characters the string can have. If you have several string variables, just list them all after the **DIM** with commas between them.

Let's store another string variable in the Computer. Don't type **NEW**. The string variable we now add is B\$.

```
5 DIM N$(9),B$(18)
30 B$="UNITED STATES"
40 PR.B$
RUN
ATARI 400
UNITED STATES
```

*Variable B\$ can be up to 18 characters long.
The LET is not needed.
The PRINT is abbreviated.*

*N\$ is still in the Computer.
B\$ is displayed.*

Do you notice the new lines?

Let's take another look at the program. We erase the screen by pressing the **SHIFT** and **CLEAR** keys together. Now type **LIST** and press **RETURN**.

```
LIST
5 DIM N$(9),B$(18)
10 LET N$="ATARI 400"
20 PRINT N$
30 B$="UNITED STATES"
40 PRINT B$
```


The original program had only three lines. (Lines 5, 10 and 20). We then added Lines 30 and 40. The **LIST** instruction tells the Computer to display the entire program.

The **READ — DATA** statements can also be used to read string variables together with numerical variables. Here's a program to find the average of three grades:

```
NEW
5 DIM N$(6)
10 READ N$,G1,G2,G3
20 DATA MARION,94,89,90
30 PRINT "STUDENT:",N$
40 PRINT "GRADES:",G1;" ";G2;" ";G3
50 PRINT "AVERAGE GRADE:",(G1+G2+G3)/3
RUN
STUDENT:           MARION
GRADES:            94 89 90
AVERAGE GRADE:    91
```

The spacing looks terrific!

A DISCOVERY

Why are the parentheses needed in Line 50?

First clue: type in `PRINT 94+89+90/3` and press **(RETURN)**.
 Second clue: type in `PRINT (94+89+90)/3` and press **(RETURN)**.

O.K. Sherlock, can you deduce the answer?

Why are there two commas after the quotes in Lines 30 and 40?

(Type the program without the extra comma and see what happens.)

BACK TO THE AVERAGING PROGRAM

How about if we now average Amy's grades? Her grades are 97, 79 (bad news), and 100 (right on!). We must change the DATA statement of Line 20.

```
20 DATA AMY,97,79,100
```

and press **(RETURN)**.

We change the program by editing or retyping Line 20 in the program. In this case, retyping is easier.

```
RUN
STUDENT:          AMY
GRADES:           97 79 100
AVERAGE GRADE:   92
```

Sure enough, the changes worked. Let's take another look at this program. Erase the screen. (Do you remember how? Yes, press the **(SHIFT)** and **(CLEAR)** keys together). Now type **(LIST)** and press **(RETURN)**.

```
LIST
5 DIM N$(6)
10 READ N$,G1,G2,G3
20 DATA AMY,97,79,100
30 PRINT "STUDENT:",N$
40 PRINT "GRADES:",G1;" ";G2;" ";G3
50 PRINT "AVERAGE GRADE:",(G1+G2+G3)/3
```

Very nice, the new Line 20 is now part of the program. The old Line 20 (with Marion's grades) is gone.

What is a sure way to live to be
100 years old?

Eat an apple a day for 36500 days.



FUN TIME — WORD SEARCH

There are 12 Computer vocabulary words hidden in the puzzle. The lucky words are always in a straight line. They may be read down or up, from left to right, right to left, or even diagonally. Circle the words. Happy hunting.

The words are:

BASIC

EQUAL

PRINT

CLEAR

LET

READ

COMMA

LIST

READY

DATA

NEW

RUN

B C D A T A O

Z A T I N E L

C Q S W S Q A

S K F I E I U

T E L T C N Q

T N I R P V E

B K R D N T A

W N T A F S M

L U M E E I M

N R J R V L O

C R E A D Y C

BRAIN FOOD

- To display the instructions of a program on the screen, type in **L I S T** and press **RETURN**.
- To change a line in the program, retype the entire line or edit the line. To delete (remove) a line from the program, type its line number and press **RETURN**.
- The DATA statement contains data for the variables in the READ statement. The DATA statement can be placed anywhere in the program, even before the READ statement.

EXPERIMENT



How can we put 2 strings together in a PRINT?

Let's see!

```
NEW
10 DIM N$(6),B$(6)
20 N$="UNITED"
30 B$="STATES"
40 PRINT N$;B$
RUN
UNITEDSTATES
```

No space between the words.

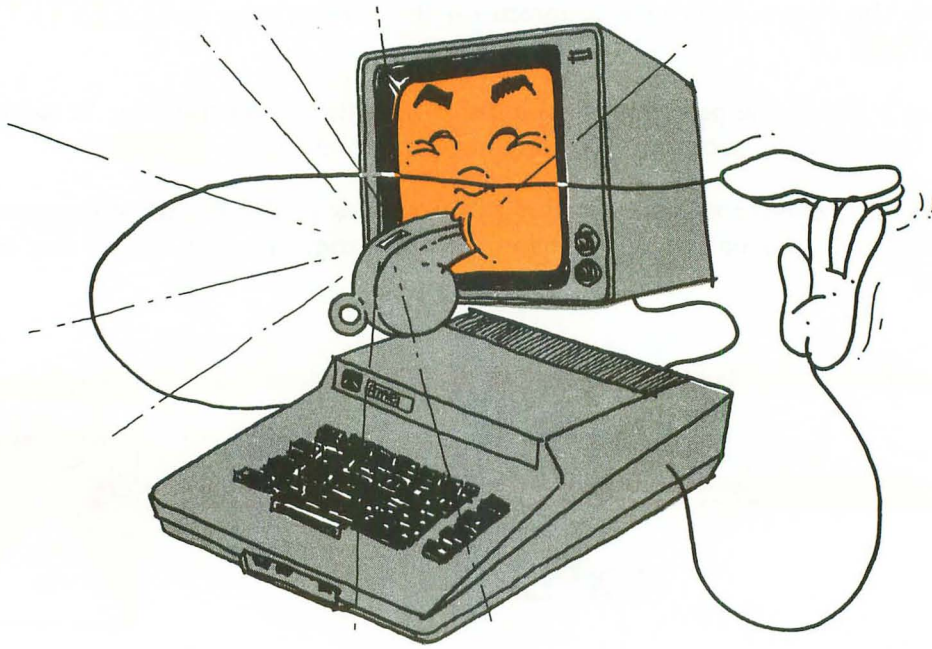
Why isn't there a space between the words? Right! We didn't put a space into the quotes. We can add a space after UNITED or we can add a space before STATES. Let's add the space after UNITED.

1. Use the **CTRL** and **↑** keys to get back to Line 20.
2. Use the **CTRL** and **→** keys to get to the end of UNITED.
3. Use the **CTRL** and **INSERT** keys to add the space, then press **RETURN**.
4. To display the two words, move the cursor (use the **CTRL** and **↓** keys) to the beginning of the RUN statement and press **RETURN**.

Did it work? The screen should now show.

```
UNITED STATES
```

The space is there — good work.



TIME OUT FOR OLD NEWS

1. There are two types of variables, numerical and string variables.
2. Names of string variables end with the symbol \$.
3. Variable names must start with a letter. The name can have 120 characters. The remaining characters can be letters or digits.
4. LET A=5 assigns the value 5 to the variable A. The LET is not necessary.
5. A\$="CARA" assigns the character string CARA to the variable A\$. The quotation marks are required.
6. Every string variable must appear in the DIM statement. DIM A\$ (10) means the string variable can be up to 10 characters long.
7. When the Computer is first turned on, all numerical variables are zero and all string variables are blank.
8. When a variable is given a new value, its old value is lost.

EXERCISES

1. Give one example for each of the two types of variables.

2. Now try this. Turn the Computer off and then back on. How will the Computer respond to each of the following instructions?

INSTRUCTION	COMPUTER'S RESPONSE
PRINT Z	_____
PRINT Z\$	_____

3. Locate errors in the following statements. Some statements are correct.

STATEMENTS	ERRORS
a. 10 READ, A, B, C	_____
b. 15 READ 13	_____
c. 20 READ X-Y, Z	_____
d. 25 DATA 5, , 6, 7	_____
e. 30 PRINT A, B; C/D	_____
f. 35 PRINTE, F. G	_____
g. 40 READ A, B, C 45 DATA 3, 4	_____

4. What display will the following programs produce? Pay special attention to the spaces (blanks) in the display.

a. NEW
 5 DIM A\$(4),B\$(13)
 10 READ A\$,B\$
 15 DATA JACK, AND JILL
 20 PRINT A\$;B\$
 RUN

b. NEW
 10 A = 6
 20 PRINT A,B,A+7;" ";A-6
 30 END
 RUN

5. Write and run a program which uses variables A, B and C to display D. Use A=2, B=3, C=4, and $D=A*B-C$.

6. Write a program to produce the following picture of a tree. *HINT: Use seven print instructions.*

```

      X
     XXX
    XXXXX
   XXXXXXX
    XXX
     XXX
      XXX
  
```



CHAPTER 6

OH — TO BE A RECORDING STAR

Writing computer programs is a lot of fun. Once we have typed them in, we can run them over and over. But what happens at the end of the day when we must turn off the Computer? We lose the program. Turning off the Computer wipes out memory. So if we do not want to have to retype the program the next day, we better save it on cassette or disk. It is like saving money in the bank.

We want to learn how to save a program on cassette tape, or on a disk, and how to load a program back into the Computer.

Later on in this chapter, we'll get to disks.

SAVING A PROGRAM ON TAPE

To save a program we must follow some simple instructions:

1. Place a cassette tape in the recorder and rewind it.
2. Set the digital counter to zero.
3. Advance the tape counter 2 numbers (never save at 0 — you won't be on the tape yet, just the leader).
4. On the keyboard type in the command

C S A V Eand press **RETURN**.

5. The Computer will beep twice (to remind you to press the **RECORD** & **PLAY** buttons).
6. Now, on the recorder press the **RECORD** and **PLAY** buttons together, then press **RETURN** again.
7. The cassette player now begins to turn.
8. Once the program is saved the **READY** appears on the screen.
9. Check the digital counter and write down where on the tape the program is saved (both the beginning & end numbers).

That's all. The program is now saved. Is it really for sure? To make sure, save the program again. But first advance the tape counter two or three numbers, then repeat steps 4 through 9.

SAVING MANY PROGRAMS ON ONE TAPE

Programs can be saved one after the other on the same tape. To avoid confusion write down the names of the programs and the digital counter readings on the case. For example:

PROGRAM TITLE	DIGITAL COUNTER
Hi-Low game	2 – 14
Star Wars	20 – 48
Arithmetic Practice	50 – 81
Shoot the Duck	85 – 106

It's fun to collect computer programs and to trade programs with your friends.

LOAD FROM TAPE TO COMPUTER

You and your friend want to play Star Wars. Place the tape containing the program Star Wars in the recorder. The program you want starts at the digital counter position 20. Rewind the tape and set the digital counter to zero. Press the **ADVANCE** button on the recorder and advance the tape to 20 on the digital counter. Type **C L O A D** and press **RETURN**. The Computer beeps once (to tell you that you are loading not saving), now press the **PLAY** button and again press **RETURN**. The cassette recorder begins to turn. When the **READY** appears, the program has been loaded. You can now play Star Wars.



CHECKPOINT

1. To save a program on tape we type the command _____ and press **RETURN**.
2. Which buttons must be pressed on the cassette recorder when we want to save a program on tape?

3. To load a program from tape type in the command _____ and press the
_____ then press the _____
button on the recorder, and press the _____ again.

FUN TIME

This is the game of Hi-Low. The Computer picks a number between 1 and 100. As you try to guess the number, the Computer tells you whether your guess is too high or too low. You continue the game until you guess the number. Here is a program for Hi-Low:

```
NEW
10 REM HI-LOW GAME
15 PR. "I AM THINKING OF A NUMBER BETWEEN 1 AND 100."
20 N = INT(100*RND(0))+1)
25 PR. "YOUR GUESS IS"; : INPUT G
30 IF N > G THEN 50
35 IF N < G THEN 60
40 PR. "***** YOU GOT IT *****"
45 END
50 PR. "YOUR GUESS IS TOO LOW - TRY AGAIN" : GOTO 25
60 PR. "YOUR GUESS IS TOO HIGH - TRY AGAIN" : GOTO 25
```

In Line 20, the Computer picks a surprise number between 1 and 100. In Line 25, you get to type in your guess. More about all that later.

Run the program to make sure you didn't make any typing mistakes.

I can always guess the number in seven guesses or less. Can you?

To save the program place a tape into the recorder and rewind it. Then reset the digital counter to zero. Advance the tape 2 numbers, and press the **RECORD** and **PLAY** buttons.

If the tape already has some programs on it rewind it and set the counter to zero. Then advance the digital counter 2 numbers past the end of the last program saved, then press the **RECORD** and **PLAY** buttons.

Type in

C S A V E

and press **RETURN**.

When the Computer has beeped twice press **RETURN** again.

CHALLENGE

Modify the Hi-Low game to count the number of guesses. Once the correct number is guessed the Computer should display the message:

**** YOU GOT IT IN ____ GUESSES ****

USING THE DISK DRIVE

When you use the disk drive you have to tell the Computer about it before you turn it on. So if your Computer is on, turn it off.

Before anything else, be sure the disk drive is connected to the Computer. The cable should be plugged into the side of the Computer and into either of the slots in the back of the disk drive itself.

Now open the door to the disk drive and turn on its power switch. That's right — you turn on the switch with no disk in the disk drive (powering up, or down, could damage the contents of the disk).

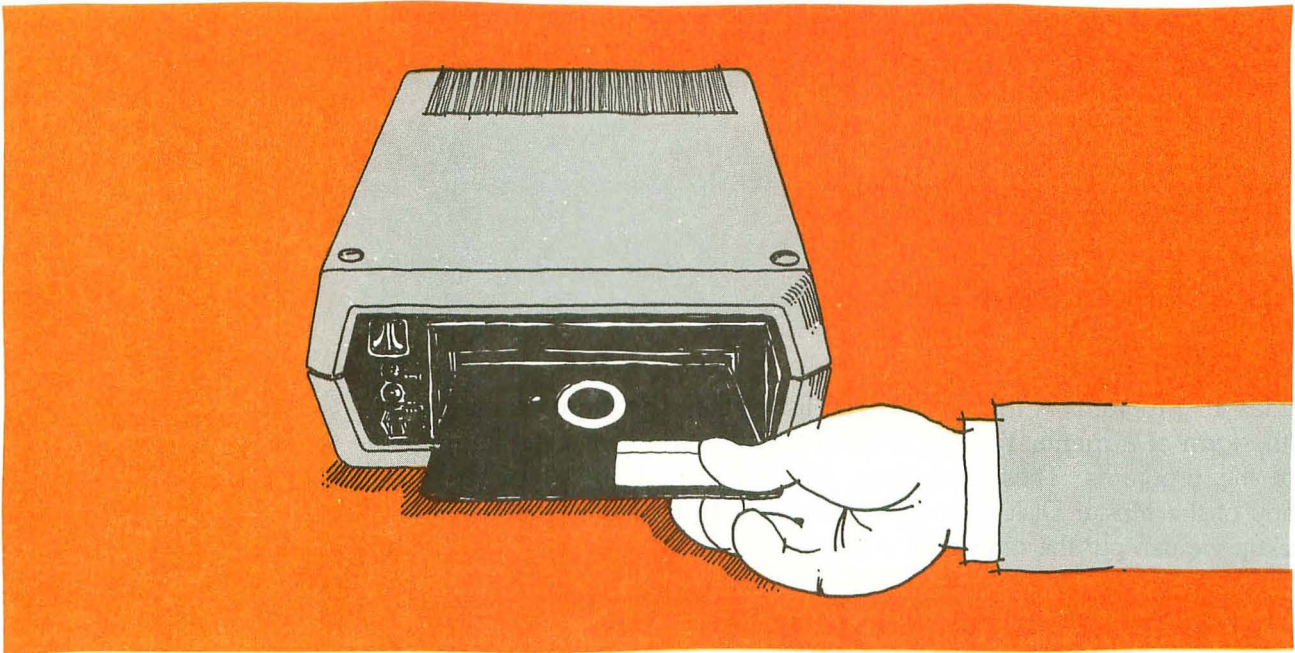


FIGURE 6.1
INSERTING A DISK

When the disk drive stops spinning put in the Master Disk (the one with **DOS** on it), close the door and turn on the Computer.

To use the disk system you have to load **DOS** into the Computer. But that's easy, just type:

D O S

and press **(RETURN)**.

D O S stands for **Disk Operating System**.

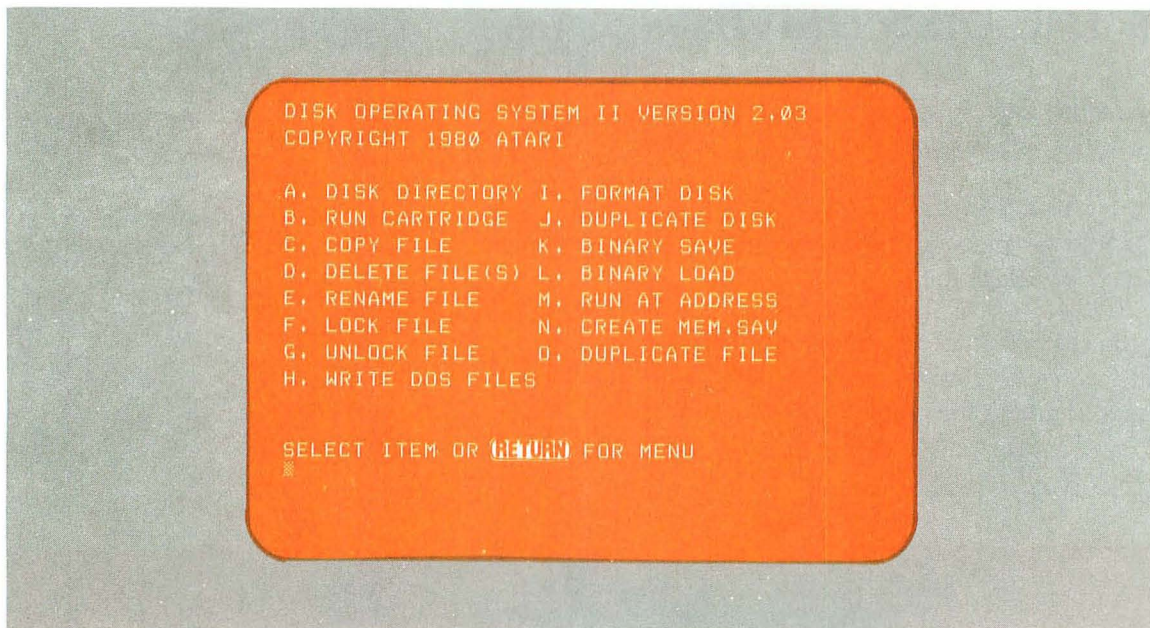


FIGURE 6.2
DOS MENU

DOS includes 15 programs A through O. The list of these programs is displayed in the form of a **menu**. Yes it is a menu because you can select and use any of the programs. These programs are very useful so we better make a copy of the Master Disk. We will backup the Master Disk. Once we have a backup we can put the original in a safe place.

COPY YOUR SYSTEM DISK

We need to use two of the **DOS** menu choices. First we need to format the new blank disk. You can also buy disks that are already formatted. These are Atari Formatted disks. Take the Master Disk out of the drive, and put in the unformatted disk. Then type:

I

and press **(RETURN)**.

The Computer responds:

WHICH DRIVE TO FORMAT?

You type **1** (if you have more than 1 disk drive put the unformatted disk in drive 1) and press **(RETURN)**. The Computer responds:

TYPE "Y" TO FORMAT DISK 1

Be sure you have the correct disk in.

So you type **(Y)** and press **(RETURN)**.

Now you want to make this a system disk, so when the formatting is complete type:

(H)

and press **(RETURN)**.

You are still using drive 1 so answer the questions the same as above.

Now you have 2 system disks — the Master Disk and your new System Disk. Put the Master Disk away in a safe place, and use it only when creating a new System Disk.

To return to BASIC type:

(B)

and press **(RETURN)**.

BRAIN FOOD

Turn on the disk drive before the Computer.

Be sure there is no disk in the drive before the drive is turned on or off.

You only format a new (unformatted) disk.

If you format a used disk you will erase all of its contents.

Keep the Master Disk safe, if you lose it you may have trouble creating a new one.

SAVING A PROGRAM ON DISK

Let's save the Hi-Low game on disk. Be sure that the disk drive is turned on, a System Disk is inserted into the drive, then the Computer is turned on. Go back a page or two to type in the game. Once you have typed in the program, run it to be sure it works.

To save the program on the disk type:

(S A V E " D 1 : H I L O W 1 ") and press **(RETURN)**.

Wow, that is fast! The **D1:** tells the Computer that the disk on which we are saving the program is in disk drive number 1.

The Computer will save the program with the name **HILOW1**. Each new program you save has its own name. Names of programs can have letters or digits or both, but:

1. the first character must be a letter, and
2. the name can be no more than 8 characters long, and
3. the name must use only letters or numbers.



CHECKPOINT

The following are names of programs. Some of these are correct names and some are incorrect. Circle the incorrect names and explain why.

"STARWARS"

"HILOW 1"

"ARITHMETIC"

"X007"

"007X"

O.K. 2 out of 5 program names are correct. The name "ARITHMETIC" is more than 8 characters long, "007X" doesn't start with a letter and the blank in "HILOW 1" is not allowed.

To load the program from disk into the Computer you type:

L O A D " D 1 : H I L O W 1 " and press **(RETURN)**.

Again we notice the **D1:**. It tells the Computer to load the program **HILOW1** from the disk located in drive number 1. The name of the program to load follows and all this information is in quotes.

Saving programs on disk or loading programs from disk into the Computer is very fast compared with tapes.

If you forget the name of a program you saved, there's no problem, type:

D O S
A

and press **(RETURN)**
and press **(RETURN)** twice.

The Computer will list the names of all the programs on that disk. This is a **directory** of the programs.

Once you've found the program you want, type **(B)** to get back to BASIC.

Then type:

LOAD "D1: program name"

and you're all set to run the program. Don't forget to re-save your program if you do change it. Use the same program name if you want the old ver-

sion of the program to be wiped out.

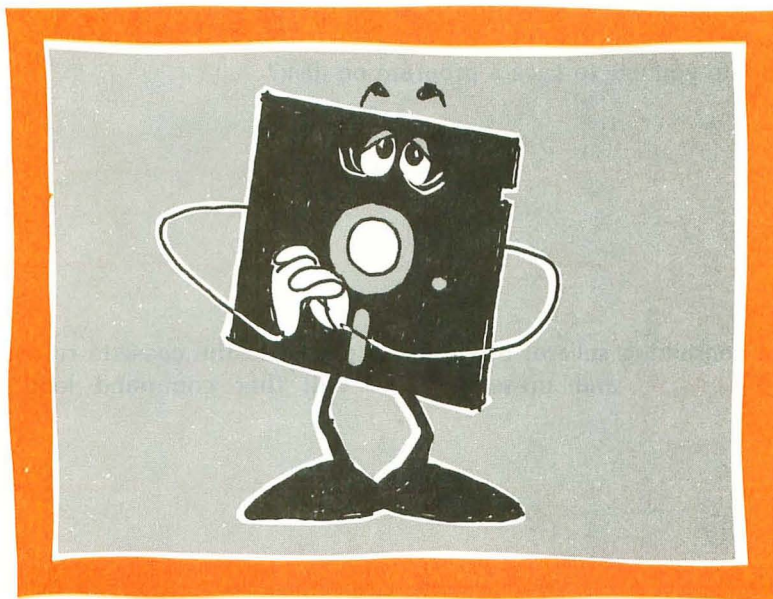
YOUR TURN

Save this program on your disk:



```
NEW
10 TONE = 0
20 SETCOLOR 2,TONE,6
25 F. X=1 TO 50 : N. X
30 IF TONE>15 THEN 10
40 TONE=TONE+1 : G.20
50 END
```

The program colors the screen in each of the available 16 colors.



CARE AND FEEDING OF DISKS

- When you handle a disk always touch only the cover — moisture from your hand will spoil the disk; scratches are a no-no.
- The disk should be in only two places — the disk drive or the protective sleeve.
- Don't bend them or force them into the drive slot.
- A cold disk needs to warm up in the room before it is used (don't heat them unless you like to see them melt).
- Keep the disk in a safe place — not near the T.V.

EXERCISES

1. Why would you want to save a program?

2. Which command do you use to save a program on tape?

3. Which command do you use to save a program on disk?

4. A rewind tape containing several programs is placed in the cassette recorder. You then type in **CLOAD** and press **RETURN**. Will that command load a program?

Which one?

5. You have a disk containing 3 programs "ONE," "TWO," & "THREE." How do you load program "TWO" into the Computer?

6. What is the difference between CLOAD and LOAD?

7. What must you do to each new disk?

8. What are the rules for a program name used to save on a disk?

9. Why might you prefer to save a program on disk instead of tape?

10. What sequence of operations must you perform before you can save a program on cassette? on disk?

a.) Cassette

b.) Disk



CHAPTER 7

A FRIENDLY CONVERSATION

Just think, you type **R U N**, press **RETURN** and the Computer displays:

```
HI, I AM TUTOR COMPUTER
WHAT IS YOUR NAME
?
```

You then type in your name **S A N T A** and the Computer responds:

```
HEY SANTA I AM PLEASED TO MEET YOU
HO HO HO
```

The **INPUT** (abbreviated **I.**) statement lets you type in information from the keyboard. When the **INPUT** statement is reached in the program the Computer stops, displays a question mark, and lets you type.

Here is the program.

```
NEW
5 DIM N$(30)
10 PR."HI, I AM TUTOR COMPUTER"
20 PR."WHAT IS YOUR NAME"
25 INPUT N$
30 PR."HEY ";N$;" I AM PLEASED TO MEET YOU"
40 PR."HO HO HO"
```

```

RUN
HI, I AM TUTOR COMPUTER
WHAT IS YOUR NAME
?SANTA
HEY SANTA I AM PLEASED TO MEET YOU
HO HO HO

```

*Request a RUN.
The Computer introduces itself.*

After the ? you type your name.

In Line 25 of the program the name you type in is assigned to the string variable **N\$**. In Line 30 **N\$** is used to display your name.

A SHORTCUT

Lines 20 and 25 can be combined:

```
20 PR."WHAT IS YOUR NAME" : INPUT N$
```

is the same as

```
20 PR."WHAT IS YOUR NAME"
25 INPUT N$
```

See the colon?

The colon combines the **PRINT** and **INPUT** instructions into a single instruction. Go ahead, retype the program with a colon and use the abbreviation **I.** for **INPUT**:

```

LIST
5 DIM N$(30)
10 PR."HI, I AM TUTOR COMPUTER"
20 PR."WHAT IS YOUR NAME":I.N$
30 PR."HEY ";N$;" I AM PLEASED TO MEET YOU"
40 PR."HO HO HO"

```



COMPUTER MAGIC

The Computer asks you to think of a three-digit number such as 222, or 666, or 999. You are then asked to input the sum of the three digits. If the number you pick is 222 then the sum you enter is 6 ($2+2+2=6$). The Computer takes this clue and figures out your three digit number. How smart!


```

NEW
10 PR. "↵"
20 PR. "THINK OF A THREE-DIGIT NUMBER"
30 PR. "ALL THREE DIGITS MUST BE THE SAME"
40 PR. "FOR EXAMPLE 777"
50 PR. "THE SUM OF THE DIGITS IS" : INPUT S
60 PR. "YOUR NUMBER IS "; 37*S

```

```

RUN
THINK OF A THREE-DIGIT NUMBER
ALL THREE DIGITS MUST BE THE SAME
FOR EXAMPLE 777
THE SUM OF THE DIGITS IS
76
YOUR NUMBER IS 222

```

```

RUN
THINK OF A THREE-DIGIT NUMBER
ALL THREE DIGITS MUST BE THE SAME
FOR EXAMPLE 777
THE SUM OF THE DIGITS IS
79
YOUR NUMBER IS 333

```

The Computer gives good instructions.

You are thinking of 222.

You type in the sum 6.

The Computer is right on.

Request another run.

You are thinking of 333.

You type in the 9.

The Computer does it again.

Pretty good! The Computer did OK twice. Type in the program and try it on a friend. It always works like magic.

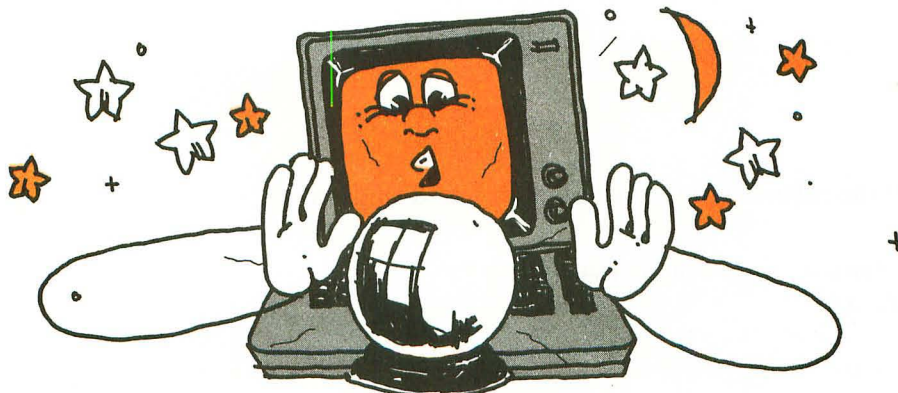
Did you notice what the **PR.** "↵" command of Line 10 did? It removed all the clutter from the screen. To clear the screen in a program you still use the **(SHIFT)** & **(CLEAR)** keys. But you use them like this:

Type **(P)****(R)****(I)****(N)****(T)****(↵)**.

Then press the **(ESC)** key once.

Then the **(SHIFT)** **(CLEAR)** combo (that gives you the **↵**).

Then type the last **(↵)**.





CHECKPOINT

1. Each of the following INPUT statements has one error. Can you find it? Circle all the errors and write the corrected statements.

INCORRECT STATEMENT

10 INPUT ,A,B

15 INPUT "WHAT IS YOUR AGE";A

20 INPUT A,A\$

25 INPUT X;Y;Z\$

30 INPUT L M N

CORRECTED STATEMENT

2. Study the two programs and fill in the missing information.

```
NEW
5 DIM A$(10)
10 PR."TYPE YOUR NAME" : I.A$
15 PR."TYPE A NUMBER" : I.A
20 PR.A$
30 PR. A,A+2
```

```
RUN
TYPE YOUR NAME
?ERIC
TYPE A NUMBER
?10
```

```
NEW
10 PR. "TYPE 3 NUMBERS"
15 INPUT X,Y,Z
25 PR. X,X+Y,X+Y+Z
35 END
```

```
RUN
TYPE 3 NUMBERS
```

```
? _____
1          3          6
```

3. Can you combine the following two-line program into a single line?

```
10 PR. "WHAT IS YOUR NAME?"
20 INPUT N$
```

THE BATTLE OF INPUT AGAINST READ

We've learned two ways of feeding the Computer.

INPUT gives you a chance to enter different information each time the program is run. That is nice, but it slows down the calculations. The Computer stops, displays a question mark and waits for you to type in information.

READ gets the information from the **DATA** statement. Each time the program is run the information is read from the **DATA** statement. It is non-stop. It is fast. If you wanted to change the **DATA** you would have to retype the **DATA** statement with different information.

Compare:

```
NEW
10 READ A,B,C
20 DATA 1,2,3
30 LET D=A+B+C
40 PR. "SUM IS ";D
50 END
```

```
RUN
SUM IS 6
```

```
NEW
10 INPUT A,B,C
      (Here there is no Line 20)
30 LET D=A+B+C
40 PR. "SUM IS ";D
50 END
```

```
RUN
?1,2,3
SUM IS 6
```

*What you type in.
The Computer responds.*

THINK

You want the sum of 4, 5, and 6. How would you do it (a) using the above **READ** program, and (b) using the above **INPUT** program?

(a) **READ** program

(b) **INPUT** program

REMARK — EVERYBODY DOES

REMark statements are placed in a program to help understand the program. **REMark**s are not displayed on the screen during a run — only during a list. If our program is already in the Computer we type **L I S T** and the program quickly appears on the screen.

LIST

```
10 REM HOMEWORK DUE JUNE 11, 2000
20 REM TEMPERATURE CONVERSION PROGRAM
30 PRINT "DEGREES CELSIUS" : INPUT C
40 F = 32 + C*9/5
50 PRINT "DEGREES FAHRENHEIT ";F
```

RUN

```
DEGREES CELSIUS
? 20
DEGREES FAHRENHEIT 68
```

You type in the 20.

The Computer gives you the answer.

Lines 10 and 20 are **REM** statements. They explain what the program does. They do not appear in the output.



EXPERIMENT

Use the temperature conversion program to draw a graph. Run the program 5 times. The first time input zero degrees for C, then 25, 50, 75, and 100. Copy the Fahrenheit degrees from the screen into the table. Then draw points on the graph. Connect the points. They form a ???.

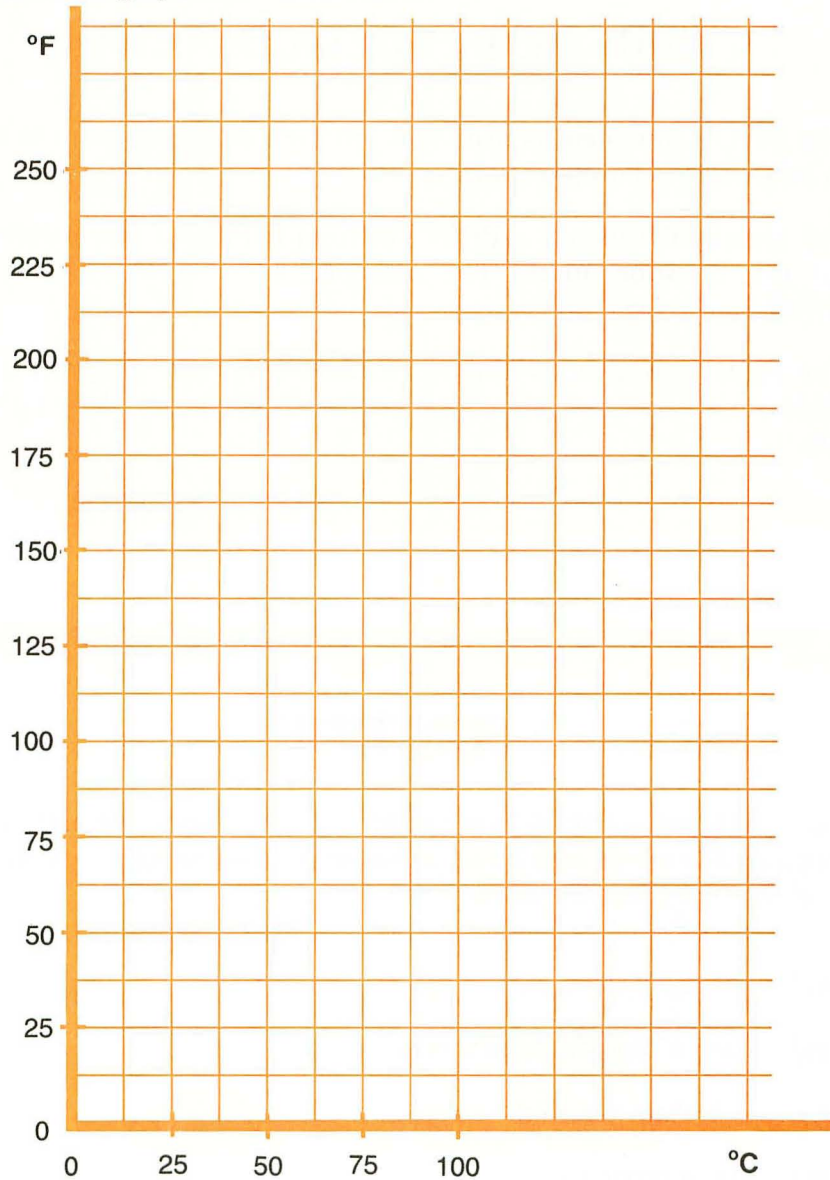
Fill in the rest of the table.

°C	0	25	50	75	100
°F	32				

TABLE

Add more points to the graph and connect them.

GRAPH



DID YOU KNOW?

Water boils at 100°C and freezes at 0°C . In degrees Fahrenheit water boils

at _____ $^{\circ}\text{F}$ and freezes at _____ $^{\circ}\text{F}$.

TIME OUT FOR OLD NEWS

The **INPUT** statement is used in a program when you want to type in information during a run.

You should always use a **PRINT** with an **INPUT** to give directions for the question mark. For example, PR. "WHAT IS YOUR NAME" : I. N\$

The **REM** statement is used to explain what the program does. **REM** statements only show up in a list. They do not show up in a run.

FUN TIME

There are 14 words hidden in the puzzle. Find the word that matches each description. The words may be up or down, from left to right, right to left, or even diagonally. Circle the words.

1. Command to display the program. _____
2. Command to load a program from a tape. _____
3. Key that erases a character on the screen. _____
4. Command to run a program. _____
5. Assigns a value to a variable. _____
6. Command to play music. _____
7. Command to store a program on a disk. _____
8. It is only sometimes required. _____
9. Contains values for variables. _____
10. Used to explain the program. _____
11. Will cause the display to appear in the next print zone. _____
12. Erases memory. _____
13. Can be abbreviated as a question mark. _____
14. Causes the Computer to display a question mark. _____

Happy hunting.

T	S	I	L	W	P	R	H	F	S	H	S	E
X	R	L	G	E	D	R	Y	S	E	N	O	A
E	T	A	T	A	D	E	T	I	N	P	U	T
N	S	P	I	N	F	E	L	N	D	S	N	E
I	D	E	F	L	P	P	R	E	M	D	D	L
E	N	K	T	H	I	T	I	N	T	P	I	N
N	E	T	H	I	L	N	R	F	G	E	R	C
B	R	A	E	N	F	I	G	R	S	W	D	N
R	U	N	N	I	E	R	D	C	T	A	T	P
E	N	E	T	P	L	P	N	M	O	U	V	L
A	D	W	T	R	I	D	N	L	O	M	B	E
K	R	T	I	E	N	D	C	L	E	M	M	E
E	A	T	R	N	E	D	L	R	P	E	I	A

EXERCISES

1. What is the difference between the PRINT“\” statement and the **SHIFT CLEAR** combination?

2. What does the following program do? _____

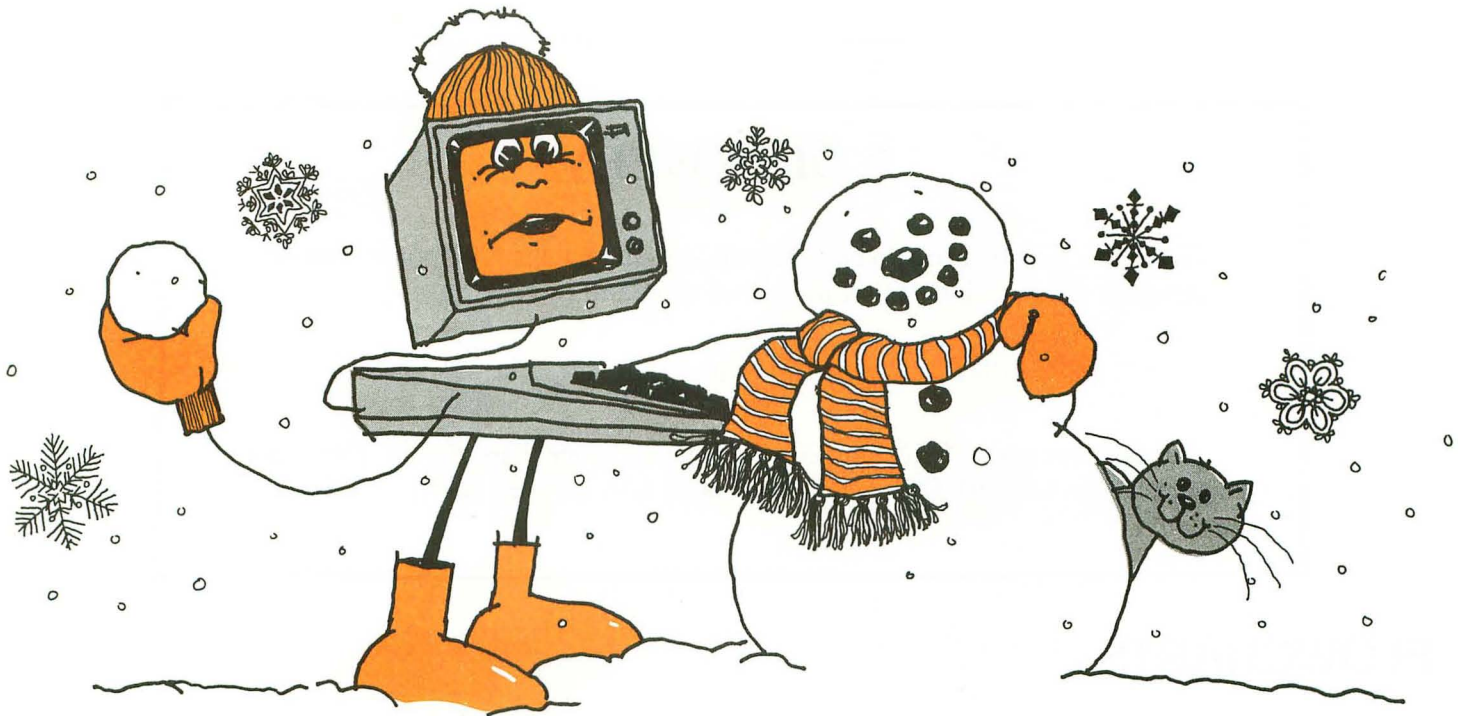
```
NEW
10 PR."WHAT YEAR IS THIS" : INPUT Y
20 PR."HOW OLD ARE YOU" : INPUT A
30 PR."IN YEAR 2001 YOU WILL BE ";2001-Y+A;" YEARS OLD"
```

3. Write a program which will produce the following conversation.

```
RUN
KNOCK KNOCK, WHO IS THERE
? GEORGE
GEORGE WHO
?GEORGE WASHINGTON
YOU MUST BE PULLING MY LEG!
```

4. Clear the screen and then list the program you wrote in exercise 3.

5. Write a program to input three numbers and display their product.



CHAPTER 8

IF YOU WANT TO

It is a cold, gray winter morning. You just woke up, and turned on the radio to get a weather report. The weatherman says:

More SNOW folks! NINE inches fell overnight.

There is no school if more than 20 centimeters of snow have fallen. Is it a snow day? Are nine inches of snow more than 20 centimeters? You turn to your Computer for the answer.

```
LIST
10 REM PROGRAM TO CONVERT INCHES TO CENTIMETERS
20 PRINT "INCHES OF SNOW" : INPUT INCH
30 REM THERE ARE 2.54 CENTIMETERS IN ONE INCH
40 CENT = 2.54 * INCH
50 IF CENT > 20 THEN 80
60 PRINT "NO SNOW DAY"
70 END
80 PRINT "YES, IT IS A SNOW DAY"
90 PRINT "      GO BACK TO BED"
100 END
```

```
RUN
INCHES OF SNOW
??
YES, IT IS A SNOW DAY
      GO BACK TO BED
```


A DISCOVERY

We notice that besides doing additions or subtractions Computers can also compare. Look at Line 50 of the above program:

```
IF CENT > 20 THEN 80
```

This line compares **CENT** to 20. If **CENT** is larger than 20, the Computer transfers to Line 80. Otherwise it continues with the next line (Line 60).

FLOWCHARTS

There is a pictorial way of showing all of this. It's called a **flowchart**. It shows the flow of the program. A flowchart for this program would be:

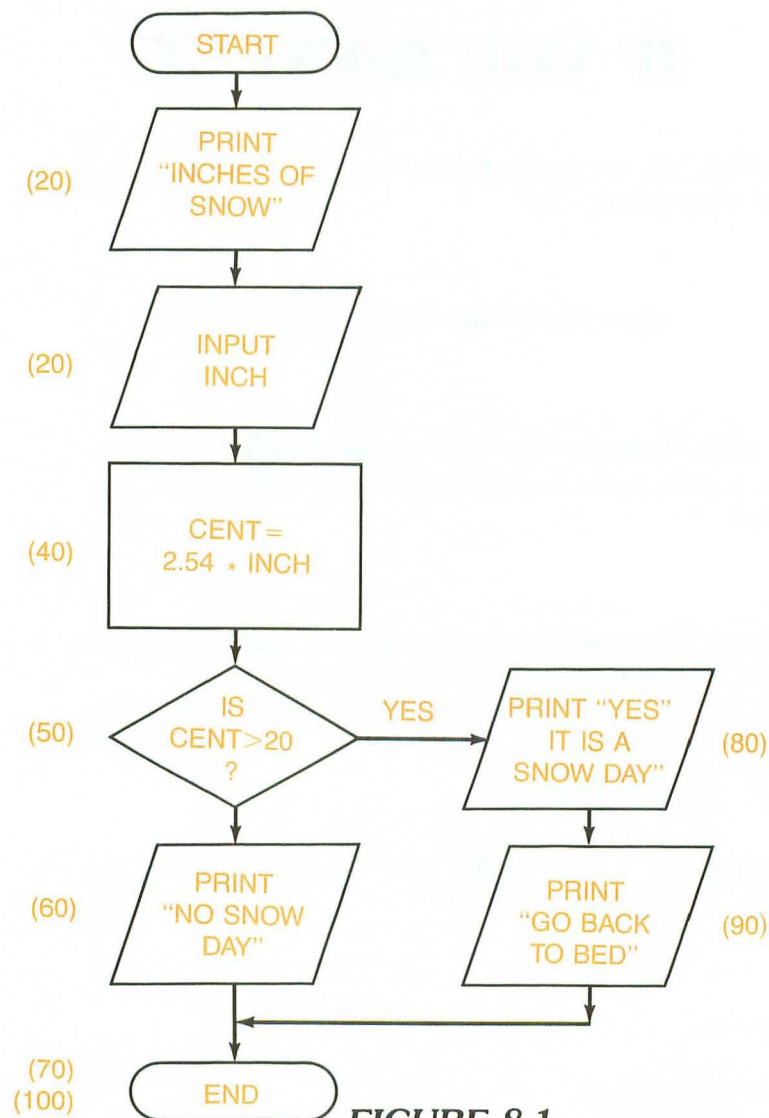
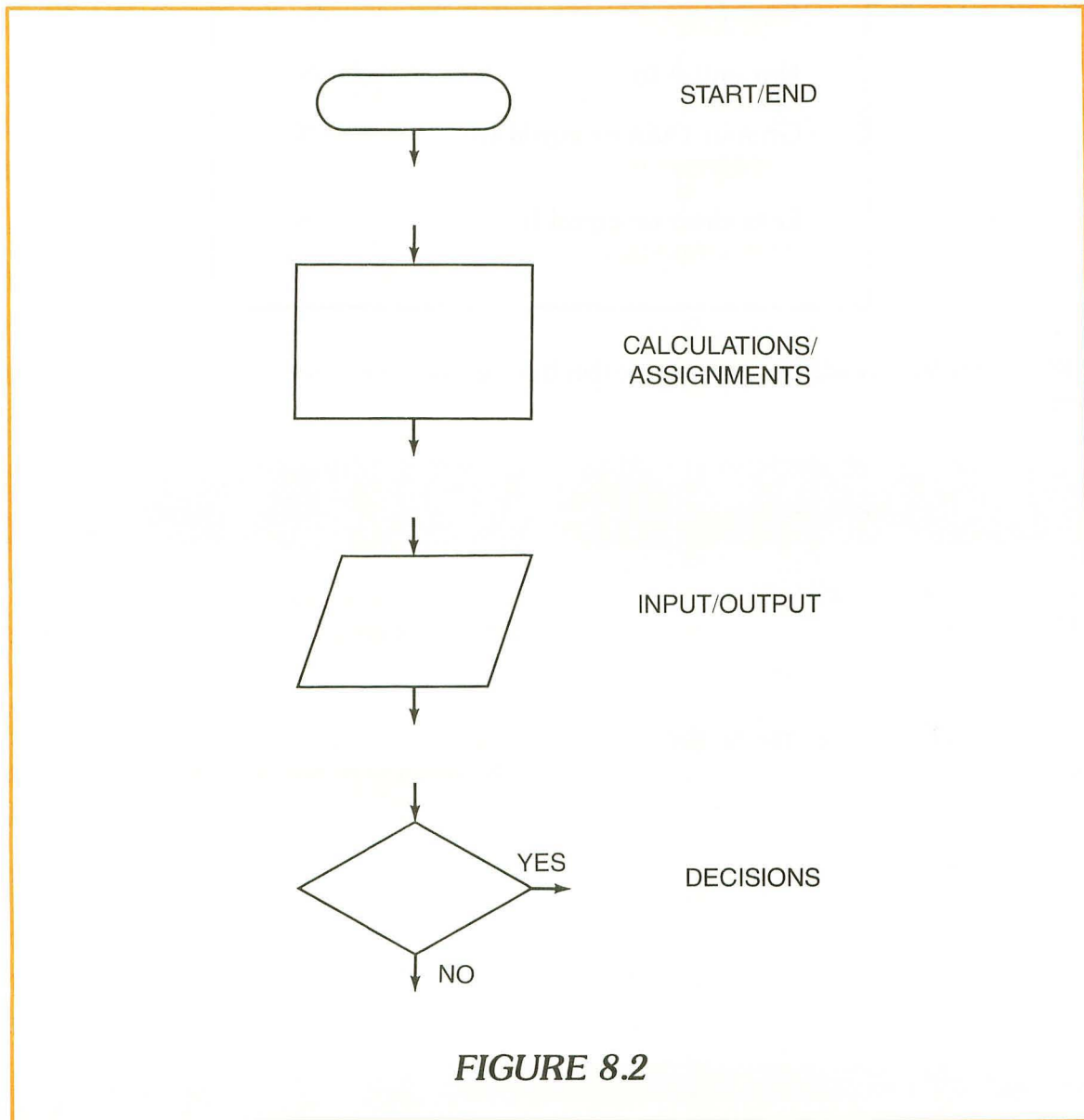


FIGURE 8.1

All information going into and coming out of the Computer is put into parallelograms (those are the **PRINT** & **INPUT** boxes above). Work is put into a rectangle (you all know what those are), and questions go into a diamond (like a ball diamond). All the boxes are connected by arrows.

Notice that the questions are the only boxes with 2 arrows coming out. They have to be labeled with either a YES or a NO (1 of each please).



Try out the flowchart for a 7 inch snow storm. Now **CENT** is less than 20 since 2.54×7 is only 17.8. So the arrow labeled NO is followed and the message NO SNOW DAY appears. Do it again for 10 inches of snow. Works pretty good, doesn't it?

Flowcharts help you see the flow of a program. Always draw the flowcharts before writing the program. Flowcharts are especially helpful when you face a problem you just can't seem to figure out. We'll try some more flowcharts later.

There are six different comparison symbols that can be used with **IF-THEN** statements.

Greater than	>
Less than	<
Equal to	=
Not equal to	<>
Greater than or equal to (Not less than)	>=
Less than or equal to (Not greater than)	<=

IF-THEN statements form conditions which are either true or false. Here are some examples:

EXAMPLES

```
10 IF A = 5 THEN 75
15 . . . . .
```

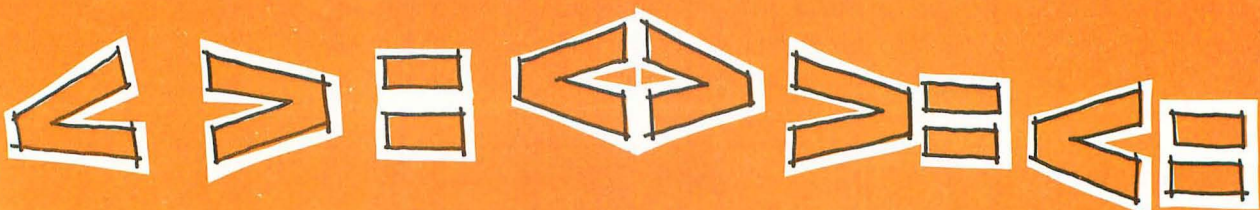
```
30 IF A+4 >= 10 THEN 80
40 . . . . .
.
.
80 . . . . .
.
.
.
```

MEANING

If A equals 5 go to Line 75.

If A does not equal 5 continue on to the next line, Line 15.

If the sum A+4 is larger than or equal to 10 transfer to Line 80, otherwise continue on to the next line, Line 40.



YOUR TURN

STATEMENT	MEANING
(fill in)	(fill in)
50 IF A <> B THEN 200	_____
60	_____
85. _____	If C is less than 1000 then transfer to Line 70, otherwise continue at Line 90.
(fill in the number)	

The first answer should be: If A is not equal to B then transfer to Line 200, otherwise continue at Line 60.

The second answer should be: 85 IF C < 1000 THEN 70
90

COMPUTER QUIZ

You enter any 2 numbers. You then ask your friend to enter the sum of the 2 numbers. The Computer checks your friend's answer.

```
NEW
10 PR."YOUR TWO NUMBERS" : INPUT A,B
20 PR."THE SUM IS" : INPUT S
30 IF S = A+B THEN 60
40 PR."NOPE, THE CORRECT ANSWER IS"; A+B
50 END
60 PR."YOU ARE A WHIZ KID"
70 END
```

```
RUN
YOUR TWO NUMBERS
?25,15
THE SUM IS
?40
YOU ARE A WHIZ KID
```

```
RUN
YOUR TWO NUMBERS
?11,13
THE SUM IS
?25
```

Request a run.

You enter 25 and 15.

For the sum you enter 40.
You are right.

Request another run.

You enter 11 and 13.

You enter 25 for the sum.

Fill in the Computer's response.



EXPERIMENT

Use the Computer quiz program to make up your own quiz.

Here are some quiz ideas:

1. Subtract, multiply, or divide two numbers. Which lines in the program need to be changed?
2. Add three numbers.

REMEMBER: To change a line in a program just use the editor.



CHECKPOINT

1. How many comparison symbols can you name? _____
2. What is the purpose of **IF-THEN** statements? _____
3. Find the errors in the following statements:

INCORRECT STATEMENTS

CORRECTED STATEMENTS

a. 10 IF A = B+12 THEN _____

b. IF 5 = C THEN 20 _____

c. 15 IF B <>= 10 THEN 25

d. 40 IF W = 10 THEN 45
45

e. 60 IF M EQUALS Z THEN 90

STRINGS AGAIN

Do you remember what a string variable is???

A string variable has a string of characters as its value. The name of a string variable always ends with the symbol \$. For example:

```
LET A$ = "I WAS BORN IN 1776"
```

Well, what is the name of the string variable we use here?

What is the character string?

MORE PRACTICE WITH IF-THEN

The **IF-THEN** statement can also be used to compare strings.

For example:

```
10 IF E$ = "YES" THEN 60  
15. . . .
```

*Transfer to Line 60 if string variable E\$="YES", otherwise
continue with the next line, Line 15.*

Which is the richest country in the world?

ANSWER:

Ireland, because its capital is always DUBLIN. (get it?)

Let's try it in a program:

```
NEW
10 DIM D$(5)
20 PR."TRUE OR FALSE 11X11=131"
30 INPUT D$
40 IF D$ = "TRUE" THEN 60
50 PR."YOU ARE RIGHT ON"
55 END
60 PR."YOU LOST YOUR COOL"
70 END
```

D\$ is dimensioned 5 because FALSE has 5 letters.

```
RUN
TRUE OR FALSE 11X11=131
?FALSE
YOU ARE RIGHT ON
```

You answer FALSE.

```
RUN
TRUE OR FALSE 11X11=131
?TRUE
```

You answer TRUE.

Now how does the computer respond?

CHALLENGE

Delete Line 55 from the TRUE/FALSE program. Remember? To delete a line, type in its line number and press **(RETURN)**.

Type **(5)(5)** and press **(RETURN)**. Line 55 has been deleted.

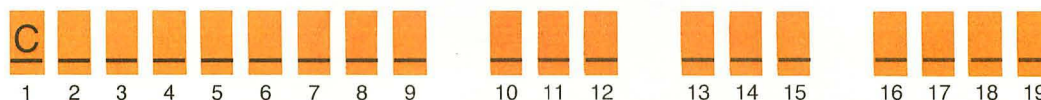
We now list the program and take a look at it.

```
LIST
10 DIM D$(5)
20 PRINT"TRUE OR FALSE 11X11=131"
30 INPUT D$
40 IF D$ = "TRUE" THEN 60
50 PRINT"YOU ARE RIGHT ON"
60 PRINT"YOU LOST YOUR COOL"
70 END
```

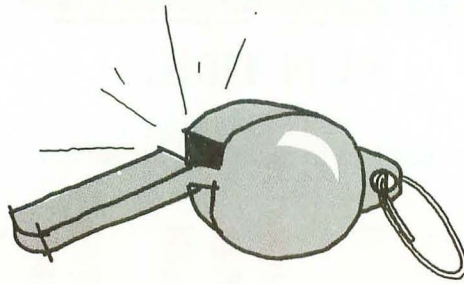
O.K., Line 55 is gone. Now what happens if we run the program? What do you find?

FUN TIME

Use the clues to figure out the sentence.



- Key to erase the screen C L E A R
1
- Abbreviation for the PRINT PRINT
2 3
- The > is a prompt
4
- <> means input
5
- The IF statement is optional.
7 6
- Nothing happens before this key is pressed ENTER
8
- B\$ is a string variable.
9
- Sign for division /
10
- Competes with INPUT PRINT
11
- It is usually optional, it stops the program STOP
12
- They go together IF THEN
13
- When the IF statement is not true, the run continues on the next line.
14
- Command to run a program RUN
15
- Used to type on the Computer TYPE
16
- Command to display a program LIST
17
- To edit a line, type its number and press RETURN.
18
- < means less than.
19



TIME OUT FOR OLD NEWS

The action has been fierce. We need some time to review.

1. The IF-THEN statement makes comparisons between numbers or between strings.

2. 10 IF A = 55 THEN 40
15

- Here the IF-THEN statement is line number 10. It is followed by Line 15.
- The condition is A = 55.
- If the condition is true the Computer branches to line number 40.
- If the condition is not true the Computer continues on the next line. That would be Line 15 in our example.

3. 10 IF N\$ = "ERIC" THEN 80

This statement compares the string variable N\$ to the character string "ERIC."

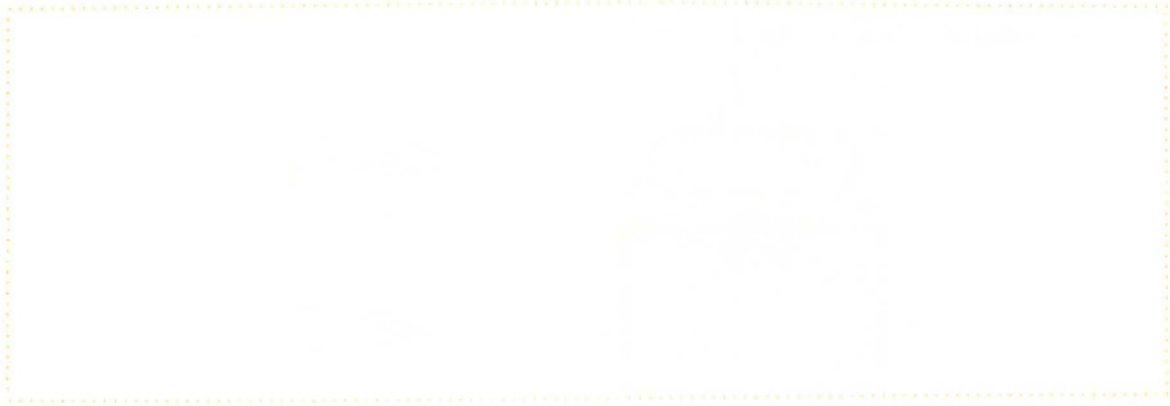
4. There are six comparison symbols:

> < = <> >= <=

5. To list a program which is in the Computer's memory we type LIST.
6. To delete a line in a program we type its line number and press **RETURN**.
7. To change a line in a program we can retype the entire line, or use the editor.

EXERCISES

1. Draw a flowchart box to branch if L is less than 22.



2. Write an IF-THEN statement to branch to Line 40 if L is less than 22.

3. Write an IF-THEN statement to branch to Line 65 if K is not equal to 63. Otherwise continue at Line 64.

4. Fill in the missing information:

10 INPUT A

20 IF A < 18 THEN _____

30 PRINT "YOU CAN VOTE"

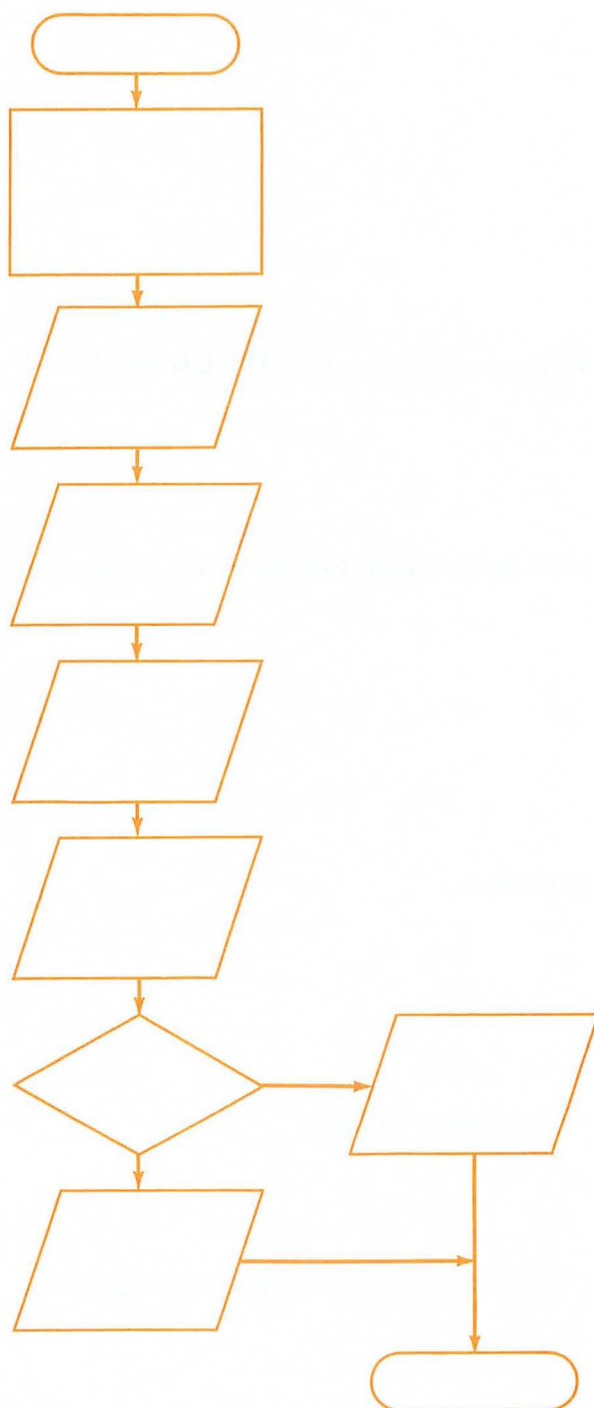
40 _____

50 PRINT "YOU ARE TOO YOUNG TO VOTE"

60 END

5. Fill these instructions into the flowchart. Note, the instructions need to be ordered.

```
PRINT "HAPPY BIRTHDAY ";N$  
PRINT "TYPE IN YOUR NAME"  
INPUT N$  
IF A$ = "YES"  
PRINT "IS TODAY YOUR BIRTHDAY "  
PRINT "TOO BAD"  
DIM N$(25),A$(3)  
INPUT A$  
END  
START
```



6. This program has two major mistakes. Can you find them and fix the program?

```
10 DIM N$(7)
15 INPUT N$
20 IF N$ = LINCOLN THEN 50
30 PRINT "NICE TO MEET YOU ";N$
40 END
```

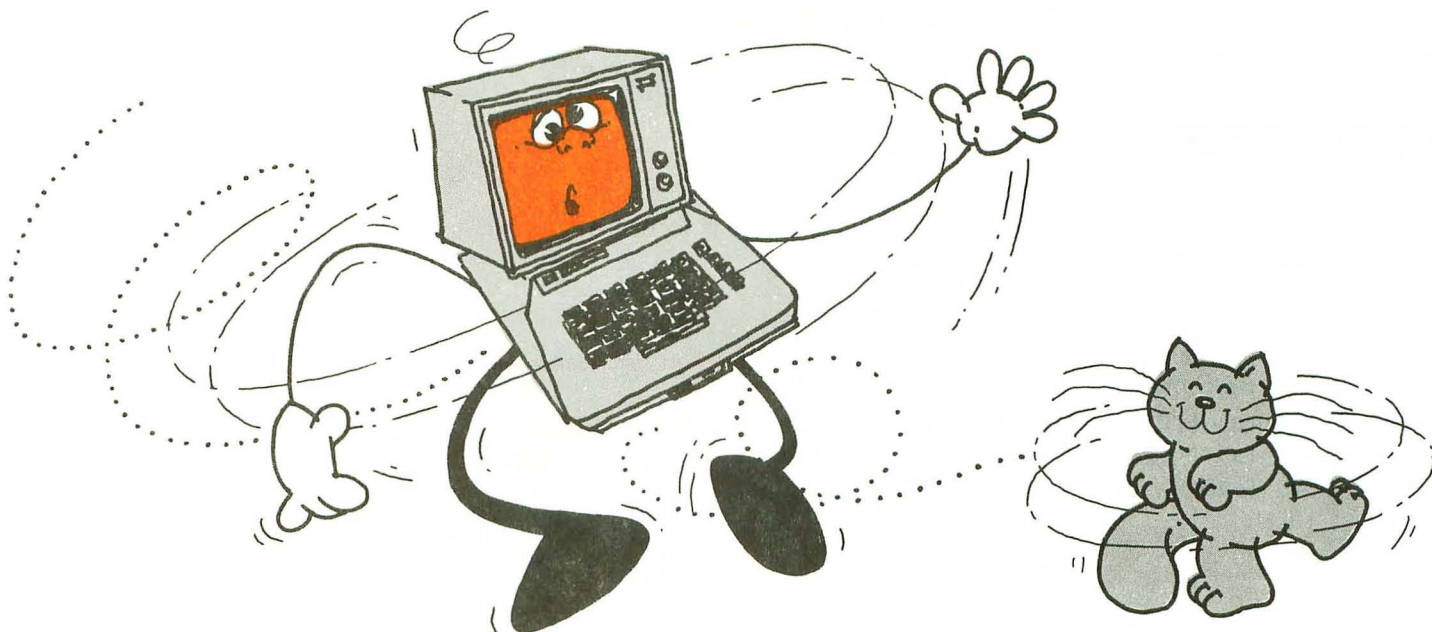
7. Draw a flowchart to INPUT a number. If the number is positive have the Computer display:

THE NUMBER _____ IS POSITIVE

Otherwise have the Computer display:

I DON'T LIKE YOUR NUMBER

8. Write a program for problem #7.



CHAPTER 9

LOOP – THE – LOOP

A Computer can do a job over and over. The Computer is accurate and fast. It does not take coffee breaks and never gets tired. Oh, I wish I could be like that.

Doing a job over and over means doing it repeatedly. We may have to do the job ten times, one hundred times or maybe a thousand times. It is nice to know that our old reliable Computer will do it for us. The job is written in the form of a computer program. The instructions that are repeated in the program form a loop. Let's try a silly loop.

```
NEW
10 REM AROUND AND AROUND WE GO
20 PR. "FOREVER"
30 GOTO 20
40 END
```

Lines 20 and 30 form a loop.

```
RUN
FOREVER
FOREVER
FOREVER
```

```
"
"
"
```

*Press the **BREAK** key to stop the looping.*

The screen is quickly filled with **FOREVER**'s. The Computer goes non-stop. The loop of Lines 20 and 30 is an **infinite loop**. The Computer prints the word **FOREVER** in Line 20. Then the **GOTO** in Line 30 sends it back to Line 20. **FOREVER** is again displayed on the screen. It will not stop by itself. But **you** can stop this madness, just press the **BREAK** key. To continue

looping type **GOTO 20** (or **G.20**) and press **RETURN** To stop it again, press the **BREAK** key.

COUNTERS

How can you make the Computer count? The best way is to use a counter. A counter looks like this:

```

NEW
10 COUNT=0
20 PR. COUNT
30 COUNT = COUNT+1
40 GOTO 20
50 END
RUN
0
1
2
"
"
```

Start the counter at 0.

Add 1 to the counter.

Go back to the print line.

In this program the variable **COUNT** is continually changing. First it stores the value 0. At Line 30, **COUNT** is increased by 1. It was zero, so it becomes 1. After that the program returns to the **PRINT**, then adds 1 to **COUNT** — now it is 2; then back to Line 20 and so on.

Let's talk about Line 30. **COUNT** is the name of a memory location in the Computer. At this line of the program the Computer takes the number stored at **COUNT**, adds 1 to it, then stores the answer back at location **COUNT**. Each time the program executes Line 30 the value of **COUNT** goes up (is incremented) by 1.

To have the Computer count by 2, change Line 30 to be **COUNT = COUNT+2**. How could the program be changed to have the Computer count by 2's starting at 100? So **COUNT** equals 100, then 102, 104, . . .

That's right — start the counter at 100. Make these two changes and list the program.

```

LIST
10 COUNT = 100
20 PRINT COUNT
30 COUNT = COUNT +2
40 GOTO 20
50 END
RUN
100
102
104
"
"
"
"
"
```

Start the counter at 100.

Add 2 to the counter.

Request a run.

THINK

Study this counter program:

```
NEW
10 COUNT = 0
25 COUNT = COUNT + 1
35 PR. COUNT
40 G. 25
50 END
```

1. What is the first number printed? _____
2. What is the second number printed? _____
3. How can you stop the program? _____

Let's use the **GOTO** to read some numbers?

```
NEW
10 READ A,B
20 PR. A;"+";B;" = ";A+B
30 GOTO 10
40 DATA 6,8,17,19,23,12
50 END
```

```
RUN
6+8 = 14
17+19 = 36
23+12 = 35
ERROR 6 - AT LINE 10
```

what happened?

Error 6 is telling you that the Computer has run out of numbers to read. This is what happens: On Line 10 the Computer reads the first 2 numbers (6 & 8). In Line 20 it prints. Line 30 tells it to go back to the **READ** (Line 10).

So . . . the Computer then reads 2 more numbers (17 & 19), prints, and returns to read the 23 & 12, print and again return. This is where the problem is — there are no more numbers to read.

Fortunately we Atari people have a way around this problem. We need to learn a new instruction called **TRAP**.

TRAP

Add this line to the program:

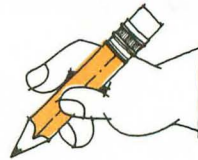
```
5 TRAP 50
```

Now list the program, and run it again.

```
LIST
5 TRAP 50
10 READ A,B
20 PRINT A;"+";B;" = ";A+B
30 GOTO 10
40 DATA 6,8,17,19,23,12
50 END
RUN
6+8 = 14
17+19 = 36
23+12 = 35
READY
```

What the **TRAP** does is to tell the Computer that if any error occurs go to the line named by the **TRAP** statement. Line 5 tells the Computer to branch to Line 50 if an error occurs. A neat way out of our problem, but don't forget that **any ERROR** in the program will spring that trap.

YOUR TURN



Make the following changes to the last program:

Replace the **READ** with **INPUT** and delete Line 40 (we don't need **DATA** with **INPUT**). Add Line 8: 8 PR. "Type in 2 numbers". Change Line 30: 30 **GOTO** 8, and delete Line 5.

O.K. clear the screen and list your program.

```
LIST
8 PRINT "Type in 2 numbers"
10 INPUT A,B
20 PRINT A;"+";B;" = ";A+B
30 GOTO 8
50 END
RUN
Type in 2 numbers
?
```

Run this program.

The Computer patiently waits at the question mark for you to type something in. How many more times will it continue to ask its question?

That's right — FOREVER.

You have to **BREAK** out of this program.

GOTO vs. IF-THEN

In the last chapter we learned about the **IF-THEN** statement, and just now we learned the **GOTO** statement. Both are **transfer statements**. Instead of going on with the next line in the program, transfer statements make the Computer branch to a different part of the program. The **GOTO** is an unconditional transfer. The **IF-THEN** is a conditional transfer.

UNCONDITIONAL TRANSFER

```
10 INPUT A,B
20 PRINT A+B
30 GOTO 10
40 END
```

*This program goes on forever. Transfer is made unconditionally from Line 30 to Line 10. To stop the run press the **(BREAK)** key.*

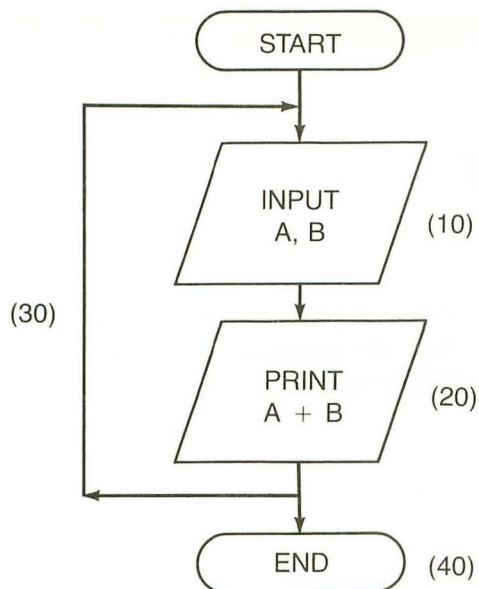


FIGURE 9.1

CONDITIONAL TRANSFER

```
10 INPUT A,B
20 PRINT A+B
30 IF A+B < 100 THEN 10
40 END
```

This program continues as long as A+B is less than 100. When A+B is not less than 100, transfer is made to Line 40 and the run stops.

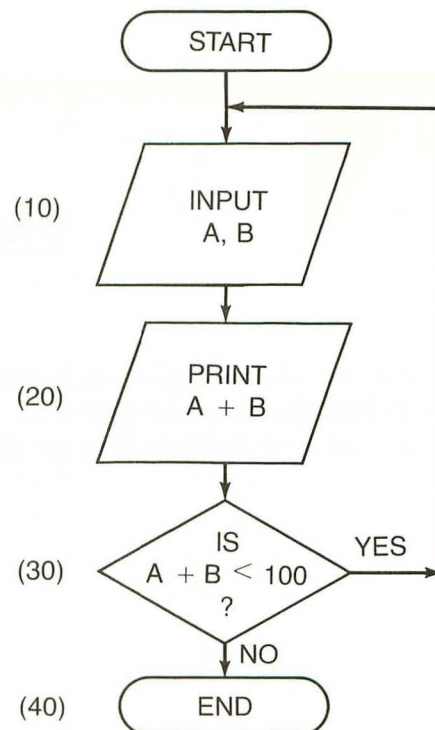


FIGURE 9.2

Take a look at this program:

```
NEW
10 PRINT "Type in 2 numbers" : INPUT L,W
20 A = L*W
30 IF A > 100 THEN 50
40 PRINT "JUST RIGHT"
45 GOTO 10
50 PRINT "TOO BIG"
55 GOTO 10
60 END
```

```
RUN
Type in 2 numbers
25,12
JUST RIGHT
Type in 2 numbers
710,20
TOO BIG
Type in 2 numbers
?
```

This combination of **IF-THEN** and **GOTO** is very common. An **IF** is usually used to do one of 2 things, go to a new line (Line 50) or continue with the next line (Line 40). The **GOTO** then loops back and starts the program again.



EXPERIMENT

Do you know why Line 45 is included in the program above? The best way to find out is to try it. Run the program with Line 45, and without it. Test the program with different numbers for L and W. Make sure $A > 100$ once and $A \leq 100$ another time.

COUNTERS THAT STOP

The counters we used earlier were endless. The only way to stop them was to press **BREAK**. What if the counter is to stop at a specific number? We may want to print all the numbers from 1 to 100.

The solution is to use a counter with an **IF-THEN** in the program.

```
20 COUNT = 1
30 PR. COUNT
40 COUNT = COUNT + 1
50 IF COUNT <= 100 THEN 30
60 END
```

Increment the counter by 1.

If the counter is less than or equal to 100 continue to loop and print, otherwise end the program.

```
RUN
1
2
3
.
.
.
99
100
```

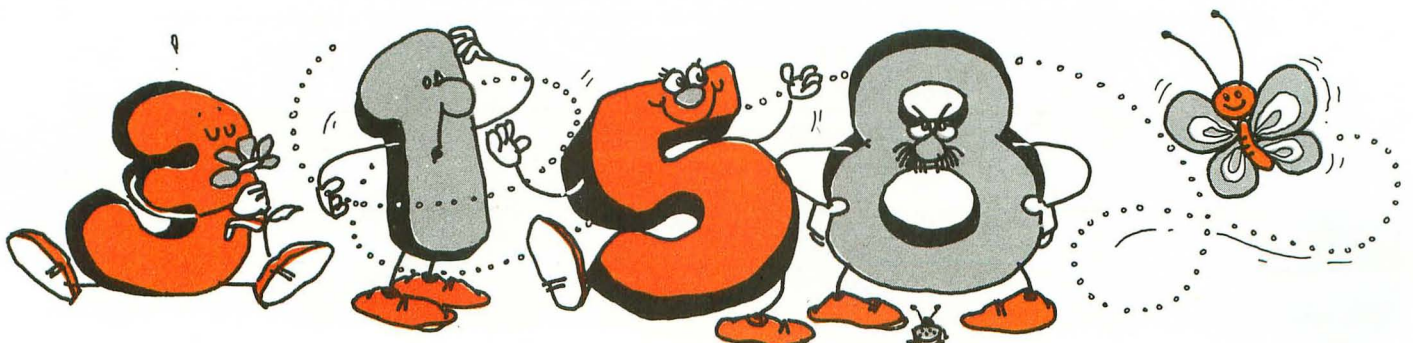
To count by 2's, simply edit Line 40:

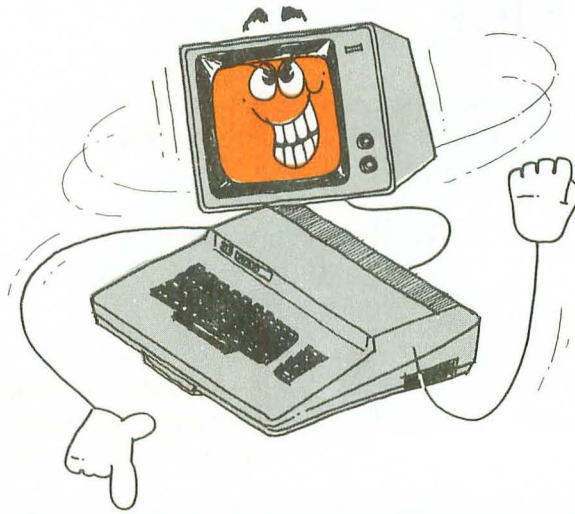
```
40 COUNT = COUNT + 2
```

Now a run will produce the odd numbers 1 through 99 — remember **COUNT** starts at 1.

```
RUN
1
3
5
.
.
.
97
99
```

The **IF-THEN** is very handy in putting a limit on loops.





CHALLENGE

There is an ancient puzzle that goes like this. A King had a problem to be solved and offered any reasonable payment for the solution. All his wise men tried and failed to solve it. Then a peasant offered to solve the problem if he would be paid in the following way. The King was to place 1 coin on the first square of a chessboard, 2 coins on the second square, 4 coins on the third square and so on, doubling the number of coins each time. The King agreed to pay and the peasant solved the problem. Write a program that will print the number of coins placed on each square of the chessboard.

TIME OUT FOR OLD NEWS

- GOTO will cause the Computer to transfer unconditionally. For example GOTO 100 will branch unconditionally to Line 100.
- IF-THEN is a conditional transfer. It will transfer to a line in the program only when its expression is true.

```

30 ...
40 IF A > 100 THEN 30
50 ...

```

If A is larger than 100 transfer to Line 30, otherwise continue at Line 50.

- TRAP is an error trap. When an error occurs in the program, TRAP will transfer to the line number in the TRAP statement.

```

60 TRAP 99

```

If an error occurs, transfer to Line 99.

Behind every successful Computer,
there stands a human being.

EXERCISES

1. What will the following programs display?

a. NEW
 10 K = 0
 20 K = K + 1
 30 PRINT K

 RUN

b. NEW
 10 Q = 0
 20 PRINT Q
 30 Q = Q + 1
 40 GOTO 20
 RUN

c. NEW
 10 DIM N\$(25)
 20 PR."What is your name"
 30 INPUT N\$
 40 PR."Hello ";N\$
 50 GOTO 30
 60 PR."GOOD-BYE";N\$
 RUN

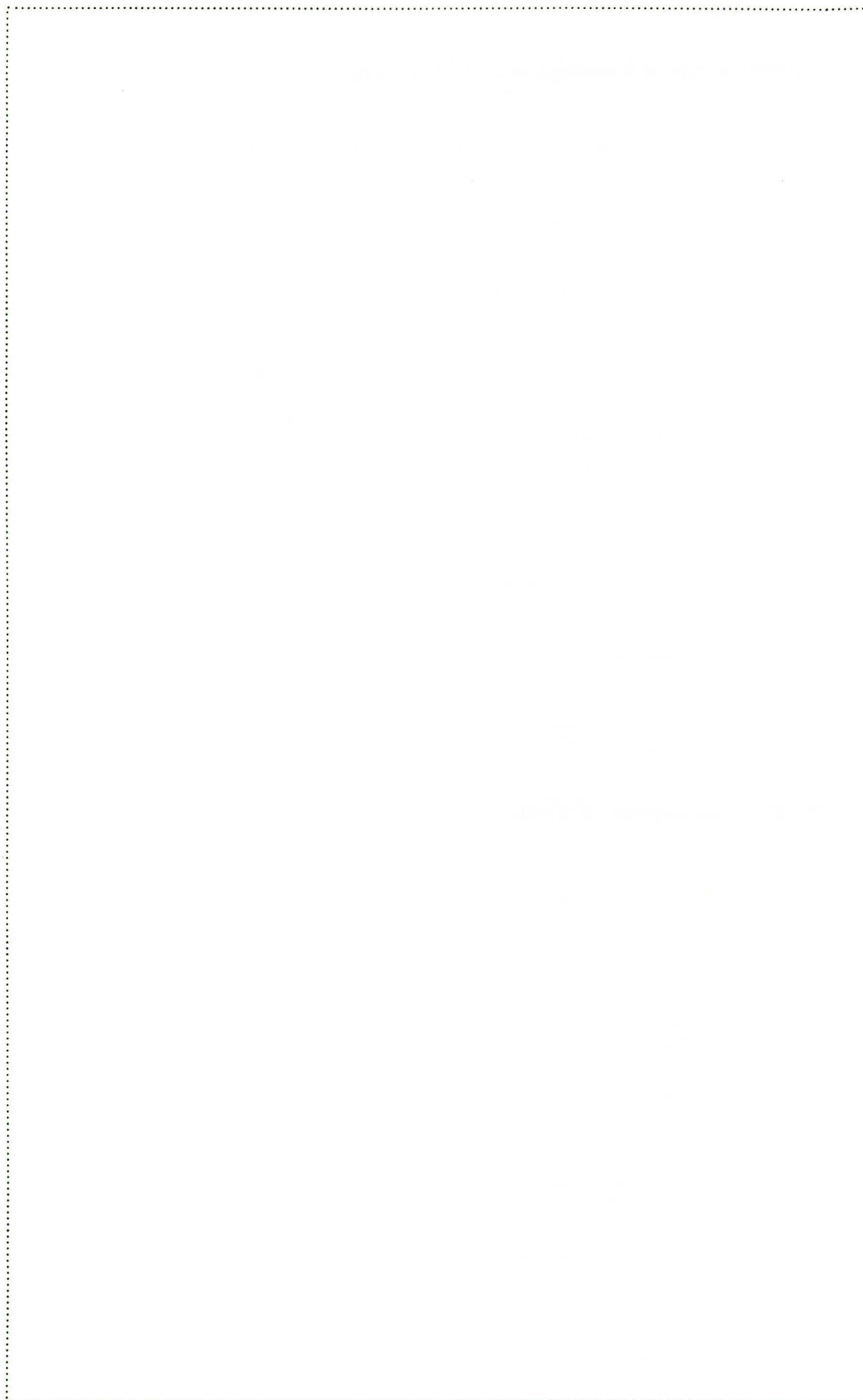
d. NEW
 10 NUM = 0
 20 PRINT NUM
 30 NUM = NUM + 2
 40 IF NUM <= 10 THEN 20
 50 END
 RUN

e. NEW
 10 K = 0
 20 K = K + 1
 30 IF K < 4 THEN 20
 40 PRINT K
 50 END
 RUN

f. NEW
 10 TRAP 60
 20 READ A,B
 30 DATA 3,6,8,16,12,24
 40 PRINT A;"+";A;" = ";B
 50 GOTO 20
 60 PR. "NO MORE DATA"
 RUN

2. Draw flowcharts for the following:

- Exercise 1b and 1d above.
- Exercise 3f(i) below.
- Read 3 sets of 4 grades and print the average of each set.



3. Write a program to:

a) Find the average of 5 numbers, over and over again.

b) Print a count-down from 100 to 1.

c) Print the counting numbers from 20 through 500 by 5's.

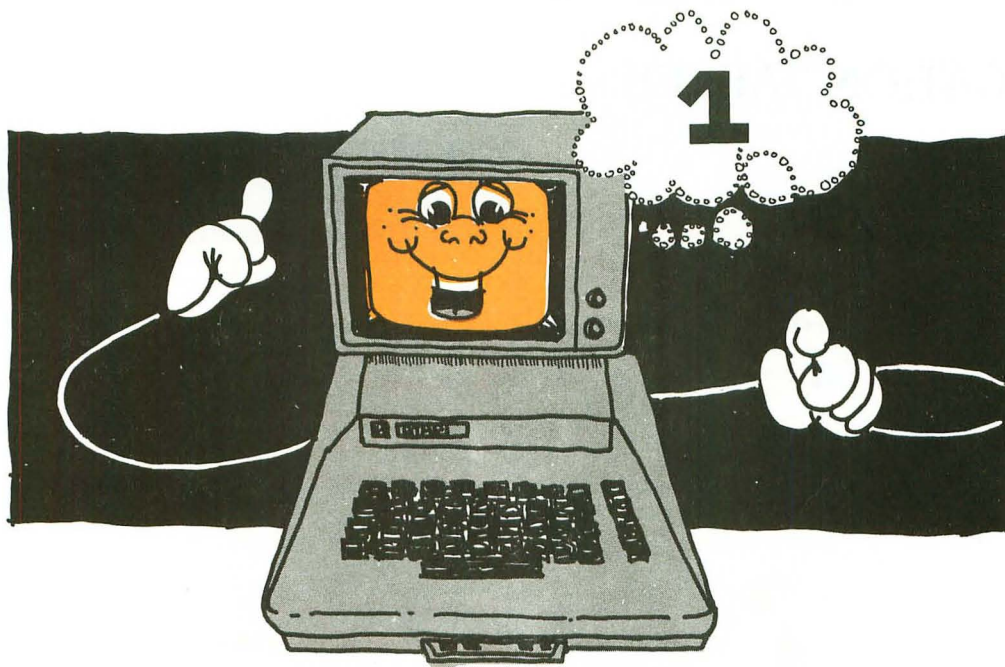
d) Modify exercise 1b above to print the even numbers 20 and over.

e) Modify exercise 1d above to print the odd numbers 11 to 27.

f) Print the numbers of years it will take for the population of rabbits on Block Island to increase from 2 to 1,000,000? Assume:

(i) The population doubles each year.

(ii) The population increases by 50% each year.



CHAPTER 10

HOW HIGH CAN YOU COUNT

The best way to form a loop is to specify how many times the loop is to be done. This is done using a **FOR-NEXT** loop. For example, suppose we want to **INPUT A** and **B**, and display **A+B** three times.

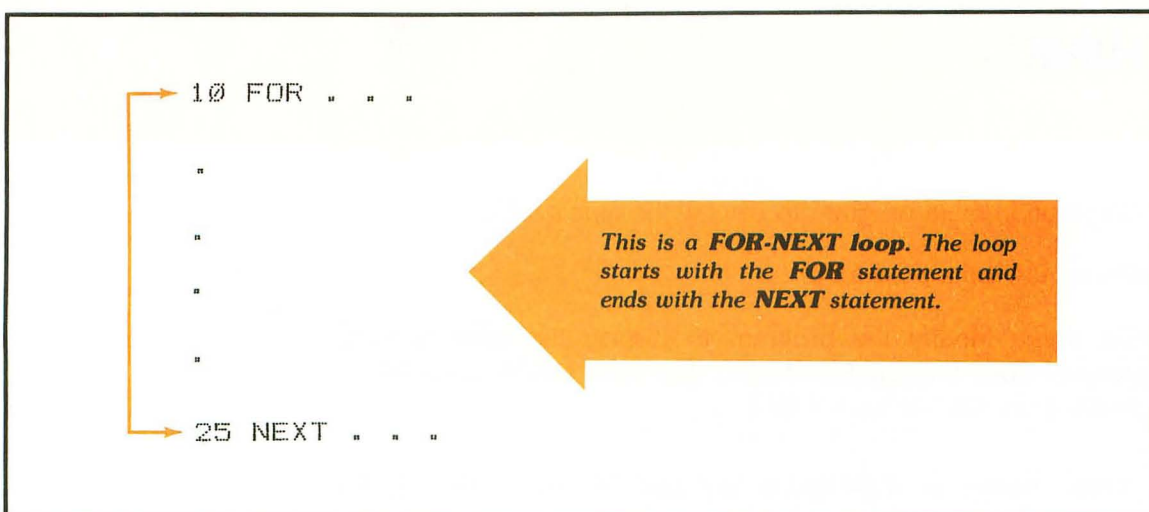
NEW

```

10 FOR N=1 to 3
15 INPUT A,B
20 PRINT A+B
25 NEXT N
30 PR."The 3 problems are done."
40 END
  
```

Set up the loop of Lines 10-25.

The first time through the loop, N=1, then N=2, and finally N=3. Each time through we enter two numbers and the Computer displays their sum.



MULTIPLICATION TABLES

HERE IS WHAT IS GOING ON

```

NEW
10 REM THE EIGHT'S TABLE
20 FOR L=1 TO 9
30 PR. L;" ";8*L
40 NEXT L
50 PR. "DONE"

```

The loop starts at Line 20 with $L=1$.
 The Computer then displays in Line 30 the value of L & 8 times L .
 From Line 40 it goes back to Line 20 and gets the next L . Now $L=2$. Then $L=3, 4, 5, \dots 9$.

```

RUN
1 8
2 16
3 24
4 32
5 40
6 48
7 56
8 64
9 72
DONE

```

First time through, $L=1$.
 Second time through loop, $L=2$.

Ninth time through loop, $L=9$.
 The loop is finished.

Pretty nifty, hey?

I meant to ask you: what is the
 oldest furniture in the world?

Multiplication tables.

THINK

Modify the multiplication table program to display the nine's table.

HINT: you must modify Line 30. Got it?

Now one more thing. Modify the program to display the table from 4 through 7 instead of from 1 through 9. *HINT: This time modify Line 20. A FOR statement does not have to start with 1.*

To make a small change in a program line use the editor (and press **RETURN**).

A DISCOVERY

What happens if you add a comma at the end of Line 30 of the eight times table program?

```
30 PR. L;" ";8*L,
```

```
LIST
10 REM THE EIGHT'S TABLE
20 FOR L=1 TO 9
30 PRINT L;" ";8*L,
40 NEXT L
50 PRINT "DONE"
RUN
```

The bracket helps you see the loop better.

Run the program; you will be surprised.

```
1 8      2 16      3 24      4 32
5 40     6 48     7 56     8 64
9 72     DONE
```

The entire table is now on three lines. Remember, there are four zones across the screen. The comma at the end of Line 30 is a **trailing comma**. It makes the Computer continue to print on the same line until all four zones are filled.

The output looks a little strange doesn't it? You would think the Computer could at least line up the numbers, but the Atari has a mind of its own. To force it to be neat add this line to the program.

```
15 POKE 82,0
```

This command makes sure that each line is displayed starting at the left edge of the screen.

then run the program:

```
RUN

1 8      2 16      3 24      4 32
5 40     6 48     7 56     8 64
9 74     DONE
```

Now replace the trailing comma in Line 30 by a **trailing semicolon**.

```
30 PRINT L;" ";8*L;
RUN
```

Run the program again.

```
1 82 163 244 325 406 487 568 649 720DONE
```

The entire table is on one line. But how do you separate the numbers?
Right — type Line 30 as:


```
30 PR. L;" ";8*L;"  ";
```

```
LIST
```

```
10 REM THE EIGHT'S TABLE
```

```
15 POKE 82,0
```

```
20 FOR L = 1 TO 9
```

```
30 PR. L;" ";8*L;"  ";
```

```
40 NEXT L
```

```
50 PRINT "DONE"
```

```
RUN
```

```
1 8    2 16    3 24    4 32    5 40    6 48
7 56    8 64    9 72    DONE
```

Notice the trailing semicolon changes the result of the POKE.

FOR-NEXT-STEP

Suppose we want to count from 1 to 9 by 2's. The **STEP** does it.

```
10 FOR M=1 TO 9 STEP 2
```

In this statement M is first 1 then (do you know? — no, not 2!) 3, then 5, 7, and 9. M is changing in steps of 2. Let's try it.

```
NEW
```

```
10 FOR M=1 TO 9 STEP 2
```

```
20 PRINT M;" ";
```

```
30 NEXT M
```

*The step is 2.
A trailing semicolon.*

```
RUN
```

```
1 3 5 7 9
```

So much for the odd numbers; now let's display the even numbers from 20 down to 10.

```
NEW
```

```
10 FOR Q=20 TO 10 STEP -2
```

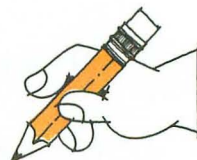
```
20 PR. Q;" ";
```

```
30 NEXT Q
```

The step -2 counts down by 2's.

```
RUN
```

```
20 18 16 14 12 10
```



YOUR TURN

Fill in the following table. Use the above program to check your answers on the Computer.

FOR Statements

Counter
VariablesValues for
Counter Variables

10 FOR I=5 TO 7

I

5. 6. 7

10 FOR L=5 TO 7 STEP 1

10 FOR O=5 TO 9 STEP 2

10 FOR V=1 TO 10 STEP 3

10 FOR E=10 TO 51 STEP 10

10 FOR M=10 TO 5 STEP -1

10 FOR Y=10 TO 5

A

101,102,103,104

T

1,1.5,2,2.5,3,3.5

A

15,10,5,0,-5,-10

R

10,8,6,4,2,0

I

.6,.7,.8,.9,1.0

CHALLENGE

Study the following program and then answer the questions. This one may be tricky.

```

NEW
10 FOR A=1 TO 10
20 PRINT "GO CARA",
30 NEXT A
40 PRINT "MARION TOO"
50 PRINT "GO! GO! GO!"

```


1. How many times will GO CARA be displayed? _____
2. How many times will MARION TOO be displayed? _____
3. How many times will GO! GO! GO! be displayed? _____
4. On how many lines will all the output appear? _____
5. Will MARION TOO appear on a separate line? _____
6. Will GO! GO! GO! appear on a separate line? _____

FUN TIME

Type in and run the following program. It may give you ideas for other programs.

```

NEW
10 PR. " "
13 PR. "COUNTDOWN"
15 FOR L=10 TO 1 STEP -1
20 PR. L
25 FOR PAUSE=1 TO 200
26 NEXT PAUSE
30 NEXT L
35 PR. "***** BLAST OFF *****"
40 FOR PAUSE = 1 TO 200
41 NEXT PAUSE
45 PR. "
46 PR. "
47 PR. "
48 PR. "
49 PR. "
50 FOR P=1 TO 16
55 PRINT " "
60 FOR PAUSE=1 TO 200
61 NEXT PAUSE
65 NEXT P
70 PR. "GONE"

```

The brackets help you see the loops better.

*A do-nothing loop.
It causes a pause.*

*The two loops are nested.
They are inside each other.*

Did you run the program?

Not bad huh? We learned two new ways to use loops:

1. The loops of Lines 25-26, 40-41, and 60-61 are **do-nothing loops**. They slow down the display on the screen.
2. Loops may be placed within loops. For example, the do-nothing loop of Lines 25-26 is inside the outer loop of Lines 15-30. The two loops are **nested loops**. Nested loops must have different counter variables names. The names of the counter variables of these two loops are PAUSE and L.

The counter variable of the outer loop of Lines 15-30 is? _____

The counter variable of the inner loop of Lines 25-26 is? _____

Is the loop of Lines 40-41 nested? _____

Does the program contain more nested loops? _____



LOOPS THAT SUM UP

We now write a program to add up all the numbers from 1 to 100. We want the sum of $1+2+3+4+\dots+98+99+100$.

To start off we'll find the sum of $1+2+3+4+5$.

```
SUM = 0
SUM = SUM+1 = 0+1 = 1
SUM = SUM+2 = 1+2 = 3
SUM = SUM+3 = 3+3 = 6
SUM = SUM+4 = 6+4 = 10
SUM = SUM+5 = 10+5 = 15
```

The sum starts out as zero.

We add 1 to the sum.

We add 2 to the sum. Now the sum equals 3.

The final answer is 15.

We can write this process as:

`SUM = SUM + N` with $N=1$, then 2, then 3, then 4, then 5.

The new value of **SUM** equals the old value of **SUM** plus N .

```

NEW
10 REM  SUM OF NUMBERS
15 S=0
20 FOR N=1 TO 5
30 S = S + N
40 NEXT N
50 PR. S
RUN
15

```

The sum S starts out at zero.

Set up a loop.

Calculate a new value for S.

End of the loop.

Display the sum S.

The sum of the numbers 1 through 5 is 15.

Variable S is the sum. In Line 15 we start the sum at zero. Then the loop of Lines 20-40 gives N the values 1, 2, 3, 4, and 5. In Line 30 the sum S is computed as the previous value of S plus N.

To find the sum of the numbers 1-100, we edit Line 20 and run the program.

```

20 FOR N=1 TO 100

```

Edit the line; now N goes to 100.

```

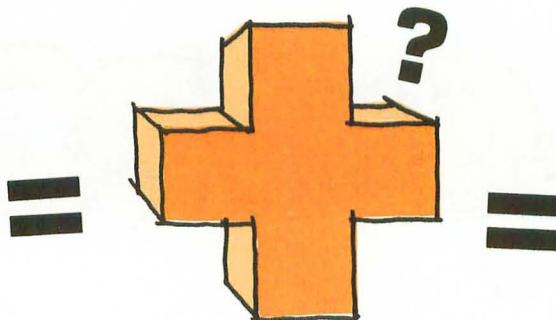
RUN
5050

```

The sum of the numbers 1 to 100 is 5050.

What is the sum of the numbers 1 through 1000?

That's right, the sum is 500,500.



IF - THEN AGAIN

I have an interesting question for you. How many numbers must we add up until the sum is just over 100?

Here we go!

```

NEW
10 REM 1+2+3+...? > 100
20 S=0 : K=0
30 K=K+1
40 S=S+K
50 IF S <= 100 THEN 30
60 PR. "THE SUM OF THE NUMBERS 1 - ";K;" IS ";S

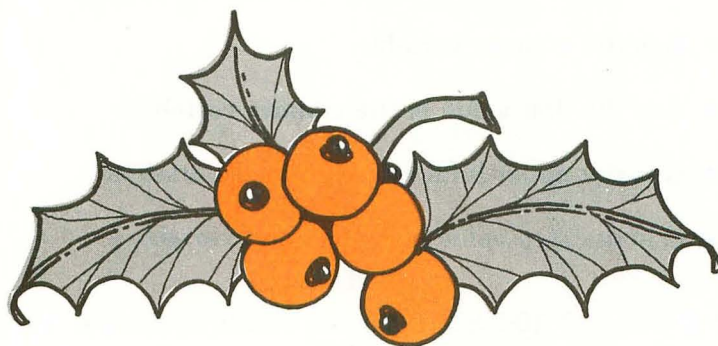
RUN
THE SUM OF THE NUMBERS 1 - 14 IS 105

```

Let's play **Computer** and go step-by-step through the program.

Program line NUMBER		Value of COUNTER K	Value of SUM S
20		0	0
30		$0+1=1$	0
40		1	$0+1=1$
50	<i>transfer to Line 30</i>		
30		$1+1=2$	1
40		2	$1+2=3$
50	<i>transfer to Line 30</i>		
30		$2+1=3$	3
40		3	$3+3=6$
50	<i>transfer to Line 30</i>		
30		$3+1=4$	6
40		4	$6+4=10$
50	<i>transfer to Line 30</i>		
	<i>and so on</i>		
30		$13+1=14$	91
40		14	$91+14=105$
50	<i>transfer to Line 60 since sum S is greater than 100</i>		
60	<i>display results.</i>		

The **IF-THEN** statement of Line 50 is a conditional transfer. As long as the sum S is not greater than 100 we transfer to Line 30. At Line 30, K is increased by 1 and at Line 40 the new sum is equal to the old sum plus K. Once the sum S is greater than 100, the results are displayed at Line 60.



THE TWELVE DAYS OF CHRISTMAS

This famous song goes like this: On the first day of Christmas my true love gave to me a partridge in a pear tree. On the second day of Christmas my true love gave to me two turtle doves, and a partridge in a pear tree. On the third day of . . . and so on for twelve days.

How many gifts were given on the twelfth day? *HINT: Use a **FOR-NEXT** loop and notice that on the first day one gift was given, on the second day 1+2 (or 3) gifts, on the third 1+2+3 (or 6) gifts, and so on.*


```

NEW
10 REM THE TWELVE DAYS OF CHRISTMAS
20 G=0
30 PR. "DAY", "GIFTS"
40 FOR D=1 TO 12
50 G = G+D
60 PR. D, G
70 NEXT D
80 END

```

*Start off gifts G=0.
Print column headings.*

Continue looping as long as D<12.

How many gifts do you think were given on the twelfth day? Run the program and find out.

TIME OUT FOR OLD NEWS

- The FOR and NEXT statements are used to form loops. The loop starts with a FOR statement and ends with a NEXT statement.

```

10 FOR K=A TO B STEP C
20
"
"
"
"
"
30
40 NEXT K

```

BODY OF THE LOOP

- K is the counter variable
- A is the first value of the counter variable
- B is the upper limit of the counter variable
- C is the step value of the counter variable
- FOR K=10 TO 100 STEP 20 will result in K=10, 30, 50, 70, 90.
- A loop without a body is a do-nothing loop. It can be used to slow down the display on the screen. For example:

```

10 PR. "START"
20 FOR PAUSE=1 TO 200
30 NEXT PAUSE
40 PRINT "FINISH"

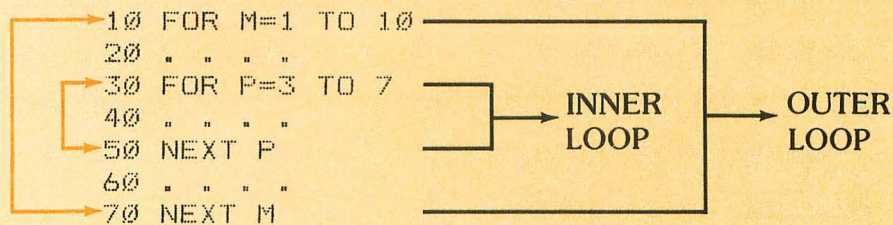
```


The word **START** is displayed first. Then after a short pause the word **FINISH** will appear on the screen.

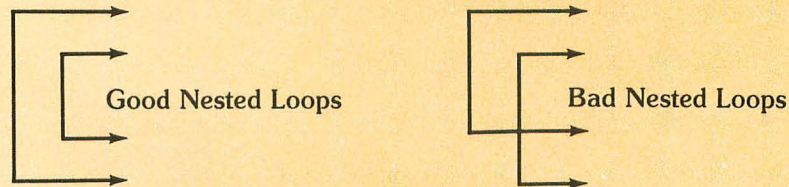
- A trailing semicolon or a trailing comma at the end of a **PRINT** statement will cause the next **PRINT** statement to display on the same line. The following three **PRINT** statements will display on one line.

```
10 PR. "JACK ";
20 PR. "and ";
30 PR. "JILL"
```

- **Nested loops** are loops within loops



Brackets show off the loops. Brackets of good nested loops cannot cross:



- A good way to check out a program is to play Computer. Go through the program step by step and write down the results. This method is especially useful when you are trying to understand a program.
- The statement $S = S + 1$ means:
The new value of S is the old value of S plus 1.
- When using a sum or a counter always give it a starting value *before* the loop. For example:

```
NEW
10 S=0
20 FOR K=1 TO 100
30 S=S+K
40 NEXT K
50 PRINT "SUM = ";S
60 END
```

S has the starting value 0.

The loop starts.

The loop ends.

EXERCISES

1. What will the following programs display?

a. NEW
 10 FOR M=1 TO 3
 20 PR. "*****"
 30 NEXT M

RUN

b. NEW
 10 FOR N=4 TO 0 STEP -1
 20 PR. N; " ";
 30 NEXT N

RUN

c. NEW
 10 FOR R=5 TO 8
 20 PRINT R, R*R
 30 NEXT R
 RUN

2. What display will this program produce?

NEW
 10 FOR K=1 TO 3
 20 PR. "ROW ";
 30 NEXT K
 40 PRINT "YOUR BOAT"
 RUN

3. Write programs to produce the following displays.

a.

```
***
***
***
***
***
***
```

b.

```
1 1 1
2 4 8
3 9 27
4 16 64
```

NEW
10 REM STARS AND DASHES

NEW
10 REM TABLE OF SQUARES & CUBES

20.

20

4. Play Computer and complete the table for the program.

NEW
10 REM NESTED LOOPS
20 FOR K=1 TO 3
30 FOR L=4 TO 5
40 PR. K,L,K*L
50 NEXT L : NEXT K

K	L	K*L
1	4	4
1	5	5
2	4	
2		

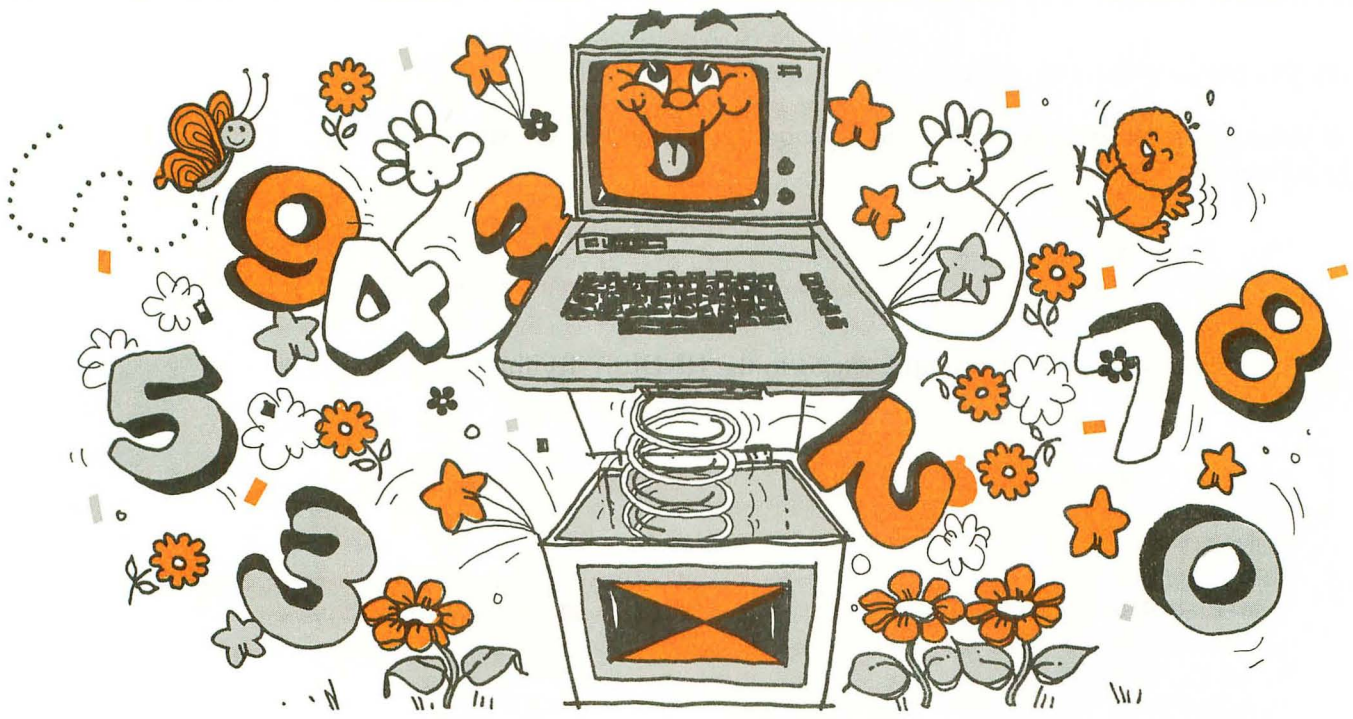
5. Write short programs to perform the following. Write these programs with an IF-THEN loop and then with a FOR-NEXT loop.

a. Display all the odd numbers 1, 3, 5, ..., 19.

b. Display all fractions $1/3$, $1/4$, $1/5$, \dots , $1/10$ in decimal form.

6. Modify the Christmas program to compute and display the total number of gifts given over the twelve days.

7. Do you remember the challenge in chapter 9 about the King and his problem? Change the program to display the total number of coins paid to the peasant.



CHAPTER 11

SURPRISE NUMBERS

When you flip a coin the toss is either heads or tails. When you toss a die (one half of a pair of dice) the outcome is either a 1, 2, 3, 4, 5, or a 6. These outcomes cannot be predicted.

The **RaNDom** number function **RND** causes the Computer to pick a “surprise” number. It is random; it is not predictable. **RND(0)** gives a number less than 1, and greater than or equal to 0. That means the number is a decimal less than 1. Try this:

```
PR. RND(0)
0.1233367919
```

The Computer's response.

If you do it 3 more times you'll get 3 different numbers:

```
PR. RND(0)
0.6681213378
PR. RND(0)
0.2716522216
PR. RND(0)
0.3979949951
```

BRAIN FOOD

The function **RND(0)** gives a number from 0 to 0.9999999999. It may equal 0 but it never equals 1.

You can also produce decimal numbers that are not between 0 and 1.

Write a statement to produce a random number between 7 and 8. We start at 7 by adding it to RND(0).

```
PR. RND(0)+7
7.5632447881
```

A random number between 7 and 8.

What do you think will happen if you add, say, 3 to the RND(0)? Try this:

```
10 FOR TRY=1 TO 10
20 PR. RND(0)+3
30 NEXT TRY
RUN
3.7431182825
3.4858398431
3.3273010296
3.1632232674
3.6084594738
3.1936492952
3.3215332075
3.6960754395
3.1949157732
3.1892359628
```

Add 3 to each random number.

*All 10 random numbers are between 3 and 4.
They are all different.*

Let's try multiplying RND(0) by 7. Edit Line 20 above and list the program.

```
LIST
10 FOR TRY = 1 TO 10
20 PR. 7*RND(0)
30 NEXT TRY
RUN
5.4376902415
3.6521366907
1.8052720882
0.2591157098
4.9970634658
2.5673807504
6.4775089903
2.1146290053
1.6680532578
5.7904543274
```

Multiply each random number by 7.

Now run the program.

All 10 numbers are between 0 and 7.

How about ten random numbers between 3 and 11? Again edit Line 20:

```
20 PR. 8*RND(0)+3
```

Add 3 so the random numbers start at 3. The difference 11-3 is 8. So multiply RND(0) by 8.

```
RUN
4.9907226576
4.5985107449
5.9635009738
5.1872558514
3.1293945365
```

All 10 random numbers are between 3 and 11.

```

6.1795654298
8.6267089815
3.7490234382
7.648437509
10.2409667953

```

Note ending 0's are not displayed.



CHECKPOINT

For each statement determine the smallest and largest possible random number.

	SMALLEST NUMBER	LARGEST NUMBER
PR. 10*RND(0)+0	0	9.9999999999
PR. 10*RND(0)+5		
PR. 10*RND(0)+10		
PR. 20*RND(0)+4		
PR. 2*RND(0)+5		

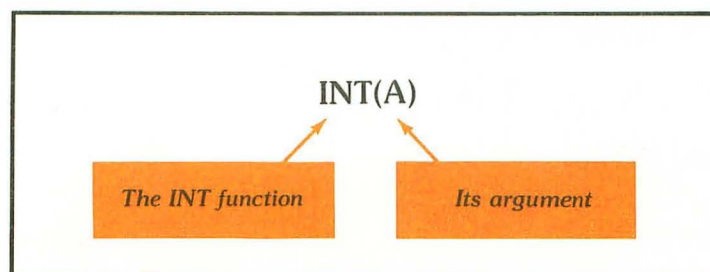
One more thing, how about getting rid of the decimals. To do that we need to learn something new, the **INT**eger function. The **INT** function, **INT()**, takes the number in parentheses and changes it to the next smaller integer.

```

PRINT INT(4.6), INT(1.1)
4          1

```

For positive numbers in the parentheses, the **INT(n)** function truncates the number n, that means it chops off the digits beyond the decimal.



EXPERIMENT

The argument in the parentheses `INT()` may be a number or an expression.

```
PRINT INT(3.14)
3
```

3 is the largest integer (whole number) not larger than 3.14.

```
PRINT INT(10*2.54)
25
```

To the nearest centimeter, 10 inches equal 25 centimeters. Here the argument is an expression.

```
PRINT INT(M/3)
0
```

M is 0 and INT(0) equals 0. The argument is again an expression.

THINK

Try to figure out the outcome of each statement.

```
PR. INT(5.2)
```

```
PR. INT(7.9)
```

```
PR. INT(0.7)
```

```
PR. INT(6+3.8)
```

```
PR. INT(7/2)
```

```
PR. INT(5/2+2.6)
```

Now let's combine **RND(0)** and **INT(n)**:

```
PR. INT(RND(0))
0
```

INT of a number between 0 and 1 is zero.


```
PR. INT (RND (0)) + 1
1
```

 $0 + 1 = 1.$

```
PR. INT (8 * RND (0))
3
```

INT of a number between 0 and 8 is a number in the range 0, 1, 2, ..., 7.

To produce a random whole number between 1 and 8 type:

```
PR. INT (8 * RND (0)) + 1
```

This is a little tough, so let's take a close look at it.

STATEMENT	GIVES	THESE NUMBERS
RND (0)	←→	0 → 0.9999999999
8 * RND (0)	←→	0 → 7.9999999999
INT (8 * RND (0))	←→	0, 1, 2, 3, 4, 5, 6, 7
INT (8 * RND (0)) + 1	←→	1, 2, 3, 4, 5, 6, 7, 8



YOUR TURN

Fill in the blanks:

STATEMENT	SMALLEST NUMBER	LARGEST NUMBER
PR. INT (10 * RND (0))	0	9
PR. INT (15 * RND (0))		14
PR. INT (20 * RND (0))		
PR. INT (10 * RND (0)) + 1		10
PR. INT (20 * RND (0)) + 1	1	
PR. INT (15 * RND (0)) + 6		
PR. INT (18 * RND (0)) + 3		
PR. INT (7 * RND (0)) + 1		

The Computer can display random numbers in any range. We can use the Computer to toss a coin. Let's say a 1 is heads and a 2 is tails.

```
PRINT INT(2*RND(0))+1
2
```

The Computer "tossed" tails.

Let's repeat this 10 times. The Computer will toss a coin 10 times.

```
NEW
10 FOR M =1 TO 10
20 PR. INT(2*RND(0))+1;" ";
30 NEXT M
RUN
1 2 1 1 2 1 1 1 2 2
RUN
1 1 2 1 2 2 2 1 2 1
```

Display a random number, a 1 or a 2.

Ten tosses are displayed.

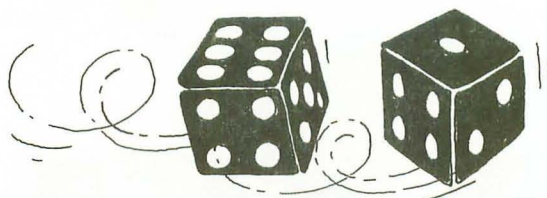
Ten more tosses are displayed.

Notice that the sequence of heads (1's) and tails (2's) is different in the two runs. They are random.

That sure is a fast way to flip a coin 10 times. Can you change Line 10 to toss the coin 100 times instead of 10 times.?

```
10 _____
```

That's right. 10 FOR M = 1 TO 100



CHALLENGE

Modify the coin tossing program to throw a **die** 10 times.

(A die is half a pair of dice.)

```
20 _____
```

In the following program a pair of dice are thrown 10 times.

The first die is tossed in Line 23. The second die is tossed in Line 26. In Line 30 the outcome of each die and their sum is displayed. Fill in the blanks.


```

NEW
20 FOR M=1 TO 10
23 X=INT(6*RND(0))+1

26 Y= _____

30 PR. _____

40 NEXT M

```

EXPERIMENT



We again randomly produce 1's and 2's. Instead of calling them heads and tails, we display a plus for a 1 and a minus for a 2.

```

NEW
10 PR. "  "
20 FOR K= 1 TO 8
25 FOR L= 1 TO 10
30 IF INT(2*RND(0))+1=2 THEN 45
35 PR. "+";
40 GO TO 50
45 PR. "-";
50 NEXT L
55 PR.
60 NEXT K
RUN

```

The random number is 1.

The random number is 2.

A blank PRINT to finish the line being used for the display.

```

-++-+---+
---+++-+
+-+---++
-++++---+
++---+---
-+-++---+
-++++---+
++-+---++

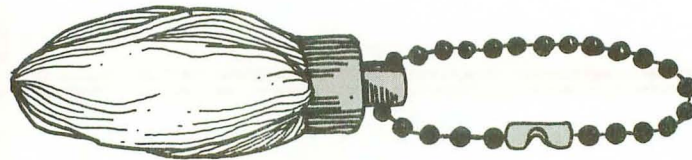
```

Type in the program. Each time you run the program the display is different. This is because the numbers are random.

Now change the symbols: \$'s for 1's and blanks for 2's. (PR. " "; displays a blank space.) Experiment with other symbols. You can also change the size of the pattern by modifying the **FOR** statements of Lines 20 and 25. And one more thing; change Line 30.

```
30 IF INT(5*RND(0))+1=2 THEN 45
```

Now there are more blanks than \$'s in the pattern. Why?



FUN WITH THE FORTUNE WHEEL

The **RND** function is used to play this game. A wheel has six lucky numbers. The wheel spins and stops at any number from 1 through 6. Can you guess the lucky number?

```
NEW
10 PR. " \ "
15 PR. "YOUR LUCKY NUMBER IS";
20 INPUT G
25 PR. "THE WHEEL IS SPINNING"
30 FOR A= 1 TO 50
35 NEXT A
40 L= INT(6*RND(0))+1
45 IF L=G THEN 60
50 PR. "THE LUCKY NUMBER IS ";L;" YOU DID NOT HAVE IT"
55 GOTO 65
60 PR. " THE LUCKY NUMBER IS ";L;" YOU GOT IT! YOU GOT IT!"
65 PR. "TO PLAY AGAIN TYPE 'RUN'"
70 END
```

Run the program 18 times. Only one number out of six is the lucky number. Therefore, you expect three guesses out of the eighteen guesses to be right on. (Get it? $3 \times 6 = 18$.) How many lucky numbers did you guess correctly?

FORTUNE WHEEL

SPIN NUMBER	MY GUESS	LUCKY NUMBER	HIT/MISS
1	5	2	MISS
2	3	3	HIT
3			
4			
5			
6			
7			
8			
9			
10			
11			
12			
13			
14			
15			
16			
17			
18			

RESULTS: I HIT _____ TIMES OUT OF 18 SPINS.

A DISCOVERY

$\text{INT}(3*\text{RND}(0))+1$ gives either a 1, 2, or 3. If the numbers are really random, then each of these numbers is likely to come up 1 out of 3 times. If we repeat it 60 times, then we expect the three numbers to occur 20 times each. Also, the average of the numbers 1, 2, and 3 is 2. Remember how to find the average of three tests? Add up the scores and divide by 3.

NEW

```

10 FOR K=1 TO 60
15 REM GENERATE A RANDOM NUMBER
20 R= INT(3*RND(0))+1
22 REM ADD UP THE RANDOM NUMBERS
25 S=S+R
28 REM TEST THE RANDOM NUMBER
30 IF X=3 THEN 60
35 IF X=2 THEN 50
38 REM COUNT ONE'S
40 A1=A1+1
45 GO TO 65
48 REM COUNT TWO'S
50 A2=A2+1
55 GO TO 65
58 REM COUNT THREE'S
60 A3=A3+1
65 NEXT K
70 PR. "ONE'S", "TWO'S", "THREE'S"
75 PR. A1,A2,A3
80 PR. "THE AVERAGE OF ALL 60 NUMBERS IS ";S/60
85 END

```

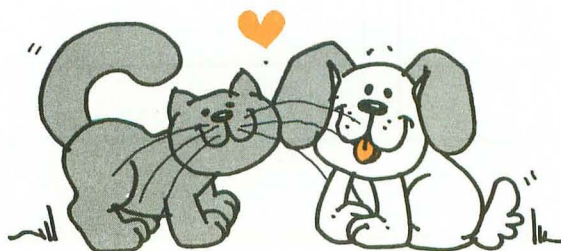
RUN

```

ONES      TWOS      THREES
17         17        26
THE AVERAGE OF ALL 60 NUMBERS IS 2.15

```

Not bad! The average is very close to the expected value of 2. Try it yourself. You'll discover that the results are different for each run, but not very different.



Run the program five times and fill in the table.

RUN NUMBER	ONE'S	TWO'S	THREE'S	AVERAGE
1	17	17	26	2.15
2				
3				
4				
5				
TOTAL				
				AVERAGE

Now let's change Line 10 to

```
10 FOR K=1 TO 300
```

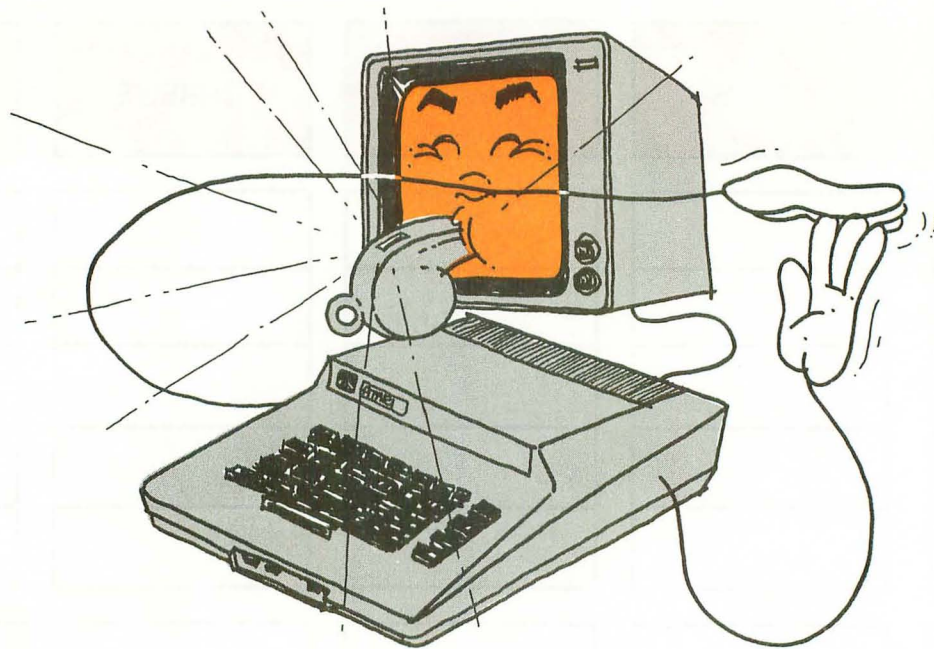
Then we also change Line 80 (fill in):

```
80 _____
```

Run the modified program five times. Is the average now closer to 2 than in the above table? Why?

**How much dirt is there in a hole
2' long, 3' wide, 4' deep?**

Answer: There's no dirt in a hole!



TIME OUT FOR OLD NEWS

- The RND function produces random numbers.
- RND(0) produces a random number between 0 and 1. It may equal 0 but never 1.

```
PR. RND(0)
0.4512349809
```

- The INT function truncates positive numbers.

```
PR. INT(7.93)
7
```

- Using INT and RND(0) together produces random whole numbers.

```
PR. INT(8*RND(0))+4
```

7 *Produces a random number in the range 4, 5, 6, ... 11.*

EXERCISES

1. Write BASIC statements to:

a. toss a coin _____

b. roll a die _____

c. display the total of two dice _____

2. Write BASIC statements to pick a number within the following ranges:

a. 0 through 5 _____

b. 1 through 6 _____

c. 1 through 100 _____

d. 5 through 20 _____

e. 7 through 8 _____

f. 0 through 2 _____

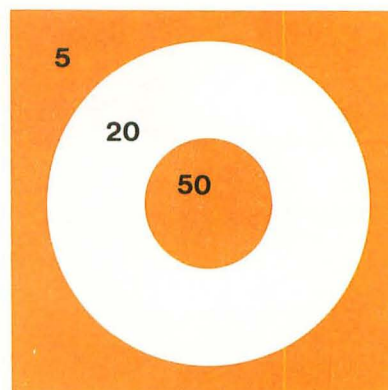
3. Write a program to toss a coin 8 times and display the sequence of HEADS and TAILS. For example, the output may be:

HEADS	TAILS	TAILS	TAILS
TAILS	HEADS	HEADS	TAILS

_____	_____
_____	_____
_____	_____
_____	_____
_____	_____

4. Write a program to toss 4 darts. A bull's eye is worth 50 points, and the other two areas on the board are worth 20 and 5 points. Keep track of your score.

Challenge a friend.



5. Write a program that asks you to predict the outcome of 5 throws of 2 dice. Each time you guess correctly, you get 50 points. Keep track of your score.



CHAPTER 12

A RAINBOW OF LETTERS

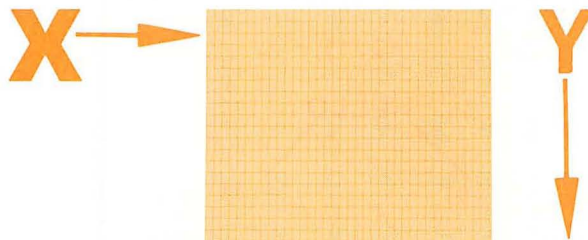
Computers are known for their ability to compute rapidly and accurately. The ATARI 400/800 series of Home Computers can also be very artistic and very colorful.

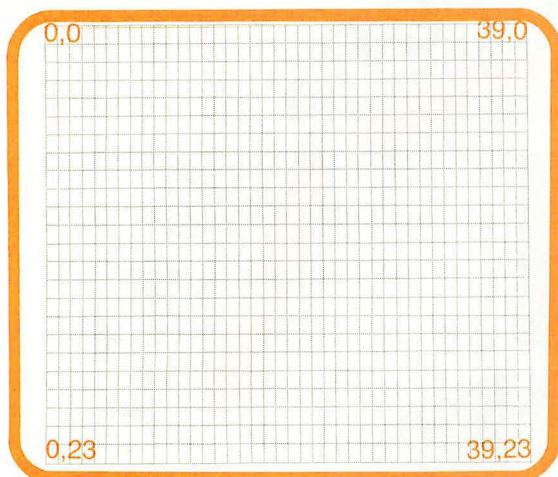
In this chapter we will learn about graphics modes 0, 1, and 2. The next chapter will discuss graphics modes 3 through 8, these modes are used to draw.

POSITION — MODE 0

When the Computer is powered up it is automatically in graphics mode 0 (GR.0).

In graphics mode 0 the screen is 40 characters wide and 24 high. The 960 (40 × 24) positions on the screen are located by using the correct combinations of numbers for the coordinates. **Coordinates** are a pair of numbers used to locate positions in Mathematics, Science, and Computer Programming. The origin, or beginning point, is called 0,0. For the ATARI Home Computer that is the corner of the screen — the upper left corner. The coordinates of the four corners of the screen for graphics mode 0 are:



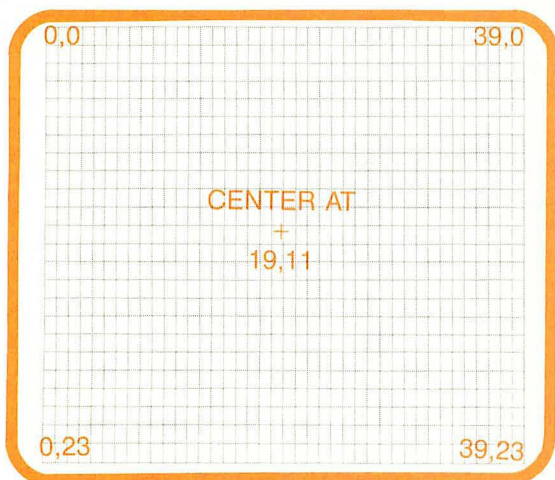


GRAPHICS MODE 0

In graphics mode 0 the screen is 40 characters wide and 24 characters high.

Let's print **HELLO** on the screen — and put it in the center of the screen. But where is the center? The center should be 20 horizontal blocks and 12 vertical blocks from the upper left corner of the screen.

The actual coordinates of the center are 19,11. Remember the ATARI Home Computer, like most Computers, starts counting at zero. That makes all the position numbers 1 less than what most people think they are.

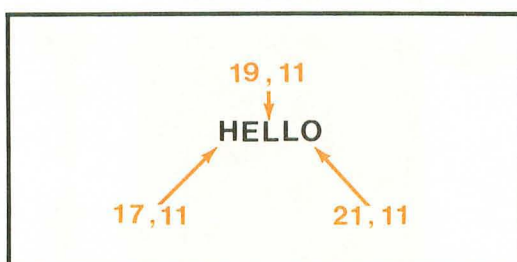


The center is at (19,11).

To place the first L at the center, the Computer needs to **POSITION** the H over 17 and down 11 (from the upper left corner). To do this, type:

```
POSITION 17,11
PR. "HELLO"
```

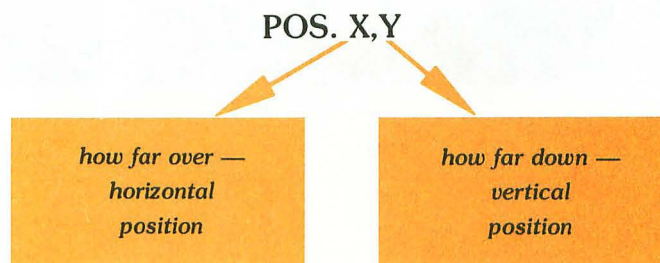
This places the cursor 17 over & 11 down. It will start to print there.



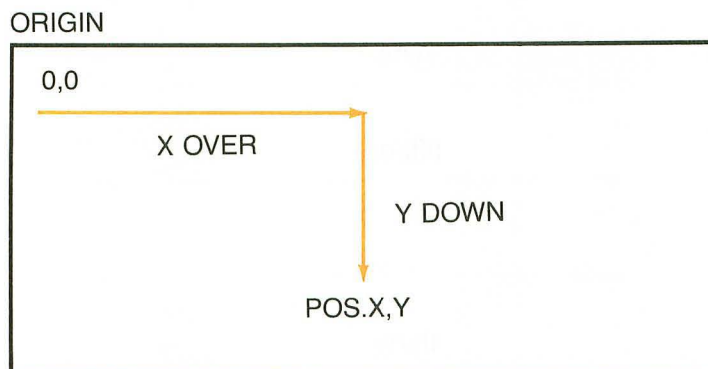
POSITION 17,11 can be abbreviated as POS.17,11.

BRAIN FOOD

POSITION just moves the cursor to a position, it does not display anything. Graphics mode 0 is 40 print positions wide and 24 positions high.



The origin is the upper left corner of the screen. Its coordinates are 0,0.



X and Y are the coordinates of the POSITION X,Y instruction.

The first **POSITION** number counts the number of columns from the left of the screen.

```
POS. 0,10
POS. 1,10
```

*The Computer uses the left most column.
The first column in from the left.*

The second **POSITION** number counts the number of rows from the top of the screen.

Any character in the top row of the screen has a second number of 0.

```
POS. 4,0
POS. 4,8
```

*The Computer uses the top row.
The eighth row down from the top.*



CHECKPOINT

Fill in the blanks in the following table:

STATEMENT	# COLUMNS FROM THE LEFT	# ROWS FROM THE TOP
POS. 4, 6	four	six
POS. 8, 15		fifteen
POS. 20, 10		
POS. _____, 4	nine	
POS. _____, 0	fifteen	
POS. 38, _____		one
POS. _____, _____	thirty	twenty-two

Try all the statements on the Computer. Don't forget to **PRINT** something at each position, for example POS. 4,6 : PR. "A"

Let's draw a line:

```

NEW
10 FOR Y=3 TO 21
20 POS. 7,Y
30 PR. "*"
40 NEXT Y

```

*The vertical position varies from 3 to 21.
Place the marker over 7 and down Y.
Print a *.
A vertical line.*

YOUR TURN

1. Display stars at positions (10,15) - (16,15) - (10,21) - and (16,21).



2. Draw a horizontal line of # characters on the 5th row.

3. Draw a vertical line of \$ characters in the 20th column.

GRAPHICS MODES 1 AND 2

Graphics modes 1 and 2 are known as text modes, they can display the alphabet and numbers only. Graphics modes 1 and 2 do not display the special graphics characters.

To shift into graphics mode 1, type:

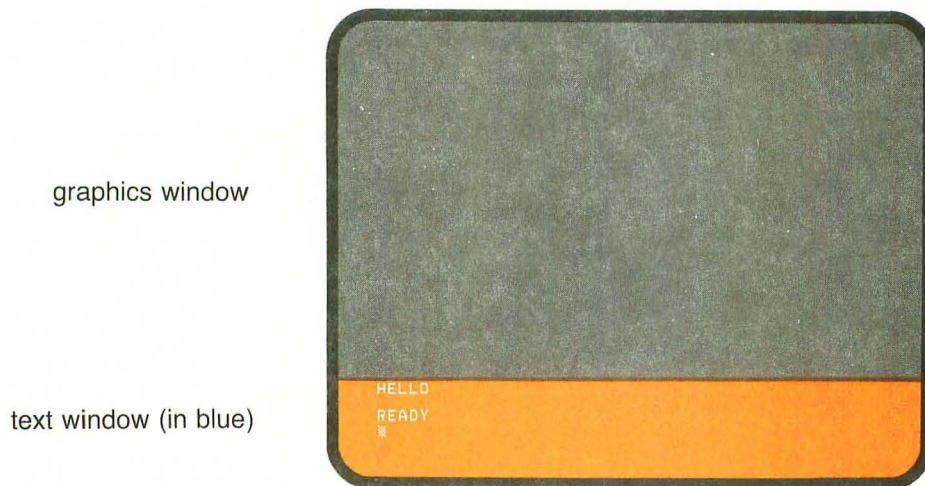
GR. 1

*This is the abbreviation for **GRAPHICS 1**.*

The screen will become black. This is the graphics window. A blue area at the bottom will light up. This is the text window. In **GRAPHICS 1**, the **PRINT** statement displays in the text window. Type:

```
PR. "HELLO"
```

The Computer responds:

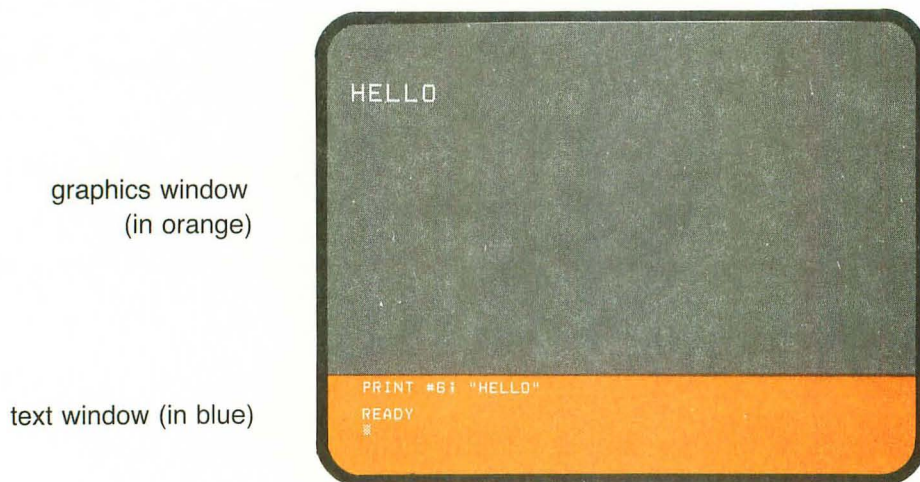


So... how can we display in the graphics window? Change the **PR. "HELLO"** to:

```
PR. #6: "HELLO"
```

*The #6 is a code for the Computer.
It means **PRINT** in the graphics window.*

The Computer responds:



Did you notice the color (orange) and the print size? Graphics mode 1 displays everything in the graphics window in double width.

Neat, isn't it?

Even better — orange isn't the only color that can be used. Type:

```
PR. #6; "hello"
```

The Computer responds:

graphics window
(in light green)

text window (in blue)



HELLO is in light green. But notice the characters are still double width capital letters. To change the color from orange to green switch the print style from upper case to lower case in the quotation marks after the **PR.#6;**.

Using print with inverse letters will also change the color of the display. Inverse letters are dark on a light background in graphics mode 0. In **GR. 1** and **GR. 2** the displayed letters are dark blue for inverse upper case and red for inverse lower case.

Remember, to get inverse letters press the Atari symbol:

BRAIN FOOD

To change the color of the characters in GR.1 and GR.2 switch from upper case to lower case, from lower case to upper case, from regular print to inverse, or from inverse to regular print. The color is selected by the style of the letter in the quotes.





EXPERIMENT

For some practice in color, type the following program.

```
NEW
10 GR.1 : DIM N$(10)
20 PR. "WHAT IS YOUR FIRST NAME";
25 I.N$
30 PR.#6;"HAVE A "
40 PR.#6;"good day "
50 PR.#6;N$
RUN
```

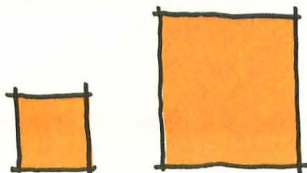
(A) is inverse A.
(day) is inverse day.

To practice **GR. 2**, edit Line 10:

```
10 GR.2 : DIM N$(10)

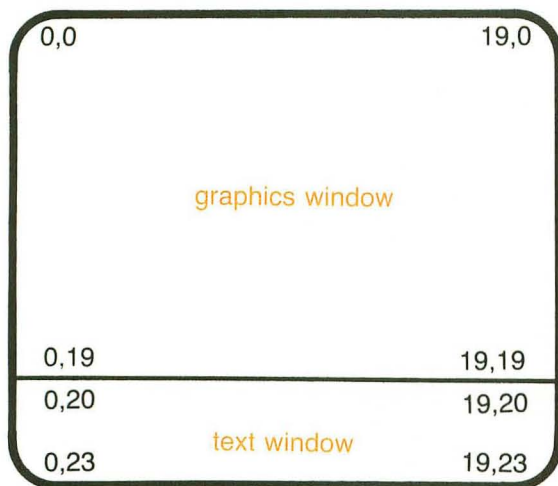
LIST
10 GR.2 : DIM N$(10)
20 PR."WHAT IS YOUR NAME";
25 I. N$
30 PR.#6;"HAVE A "
40 PR.#6;"good day "
50 PR.#6;N$
```

The difference between the graphics modes 1 and 2 is the height of the characters. **GR.2** displays its letters and numbers twice as tall as modes 0 and 1, but keeping the width of **GR.1**. That makes **GR.2** characters double wide and double tall.



POSITION - MODE 1

The coordinate system of **GR.1** is not as wide as in mode 0. Each character displayed is double the width of those in **GR.0**. That means there are half the positions across the screen - 20. There are still 24 rows, but the last 4 are used by the text window. The screen's corner coordinates are:



GRAPHICS MODE 1

The graphics window is 20 lines high.

The text window is 4 lines high.

A **POSITION** instruction before a **PRINT#6;** will display the first character of the **PRINT** at the given coordinate. If the coordinate falls in the text window, the **PRINT**ed word will not be seen. The text window hides it.

The text window can be eliminated in graphics mode 1 by typing **GR.1+16** (or **GR.17**) in place of **GR.1**. But be careful, when the Computer has finished a program and prints **READY** it needs a text window. To overcome this problem don't use **END** in the program, use a **GOTO** as in Line 40 below.

EXPERIMENT



Let's display a * in the center of the screen in graphics mode 1.

```
NEW
10 GR.1+16
20 POS.9,11
30 PR.#6;"*"
40 G. 40
RUN
```

No text window.

Place a marker at the screen's center.

*Display a * at the marker.*

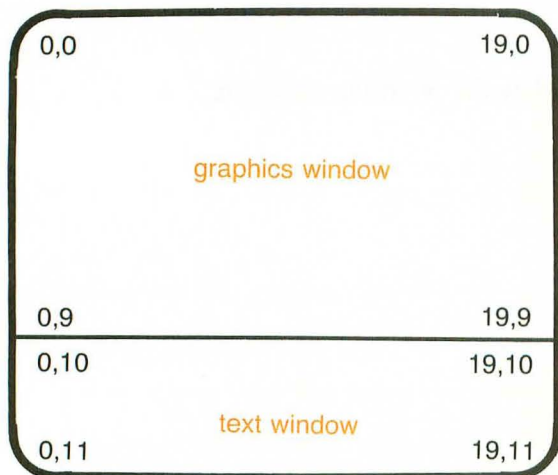
*Don't print **READY**. A trick - the Computer loops on itself.*

1. Modify this program to print a * above and below the * in the program.

2. Again modify the program to print a * to the left and the right of the * displayed on the screen.

POSITION - MODE 2

In graphics mode 2, the graphics screen is 20 columns wide and 12 rows high. The last 2 rows are used by the text window. As in **GR.1**, the text window can be eliminated by using **GR.2 + 16** (or **GR.18**) instead of **GR.2**. The screen's coordinates are:



GRAPHICS MODE 2

The graphics window is 10 lines high.

The text window is 2 lines high.

CHALLENGE

NEW

10 GR.2

→ 20 FOR DOT = 0 TO 9

30 POS.DOT,DOT : PR.#6;"#"

40 POS.19-DOT,DOT : PR.#6;"#" 19-DOT is an expression.

→ 50 N.DOT

60 END

RUN

Double size characters.

Change the PRINTing positions.

It is used for the first coordinate of POS.. When

DOT is 9, 19-DOT is 10.

Not bad, a V of #'s. It's fun to display different patterns.

How can the program be changed to display the 2 halves of the V in two different colors?

TIME OUT FOR OLD NEWS

- POSITION places the cursor at any position on the screen.
- GRAPHICS 0 is all text window - 40 columns by 24 rows.
- GRAPHICS 1 displays capital letters in its graphics window - twice the width of letters in graphics mode 0.
- GR. 1 has a graphics window 20 columns by 20 rows and a text window 20 columns by 4 rows.
- GRAPHICS 2 displays capital letters in its graphics window - twice the width and twice the height as those in graphics mode 0.
- GR. 2 has a graphics window 20 columns by 10 rows and a text window 20 columns by 2 rows.

- Both GR.1 and GR.2 can display capital letters and numbers in 4 different colors:

<u>STYLE OF CHARACTER</u>	<u>COLOR DISPLAYED</u>
upper case letters & numbers	orange
lower case letters	light green
inverse upper case letters & inverse numbers	dark blue
inverse lower case letters	red

- PRINT#6; will display characters in the graphics window in GR. 1 & 2.
- PRINT will always display characters in the text window.

EXERCISES

- Complete the following table with the expected results. Check your answers on the Computer.

STATEMENT	EXPECTED RESULT
a. PR. " ↵ "	
b. POS. 9, 9 : PR. "HI"	
c. POS. 9, 9 : PR. " "	
d. GR. 0 : PR. "HI"	
e. GR. 1 : PR. "HI"	
f. GR. 2 : PR. "HI"	
g. GR. 1 : PR. #6; "HI"	
h. GR. 2 : PR. #6; "hi"	

2. For each of the following give the coordinates of the 4 corners of the screen, the graphics window and text window.

CORNERS OF
ENTIRE SCREEN

CORNERS OF
GRAPHICS WINDOW

CORNERS OF
TEXT WINDOW

a. GR. 0

b. GR. 1

c. GR. 2

d. GR. 17

e. GR. 18

3. How do you change the colors of the characters in **GR.1** and **GR.2**?

4. How is the text window eliminated in **GR.2**?

5. Show the output for the following programs:

a.. NEW

```
10 PR. "  "
20 FOR A=1 TO 200
25 NEXT A
30 POS.9,7
40 PR. "DANGER AHEAD"
50 GOTO 10
```

b.. NEW

```
10 GR.1
20 PR."HELLO"
25 F. A=1 TO 500 : N.A
30 POS.6,3 : PR.#6;"HELLO"
40 F. A=1 TO 500 : N.A : GR.2
50 POS.6,3 : PR.#6;"HELLO"
60 END
```

6. Change the programs in #5 above to do the following:

a. Have the display stay on the screen for double the time allowed in exercise 5(a).

b. Print **HELLO** in the text window, **GRAPHICS 1**, then **GOODBYE** in the graphics window.



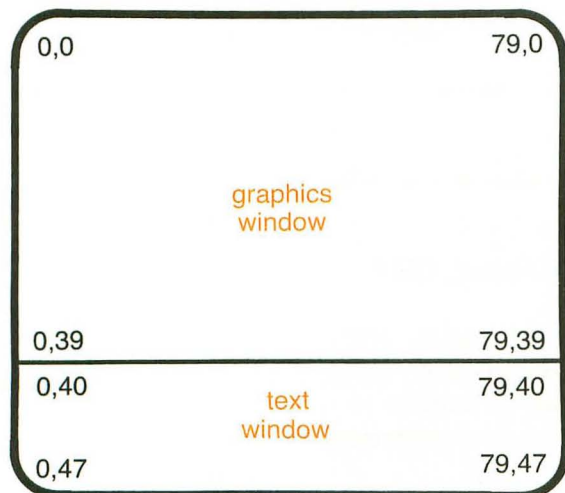
CHAPTER 13

COLOR ME PURPLE

Chapter 12 explored the three graphics modes, 0, 1, and 2. They make printing artistic. The other 9 graphics modes, numbered 3 through 11, are used to draw. In this chapter we explain graphics modes 3 through 8.

GRAPHICS MODES 4 AND 6

Graphics modes 4 and 6 are two-color modes. The display characters are in one color and the background can also be only one color. This is not as colorful as some other modes, but it requires less memory. That may be important if the program is long.



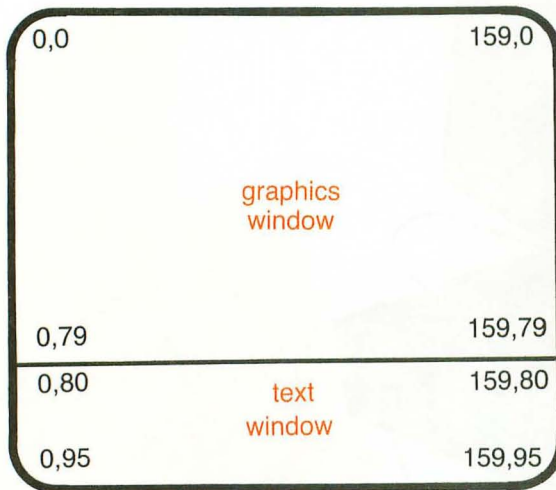
GRAPHICS MODE 4

These are the corner coordinates of graphics mode 4.

GR.4 uses a graphics window that is 80 columns and 40 rows.

As with GR. 1 and GR.2 the bottom of the screen has a text window.

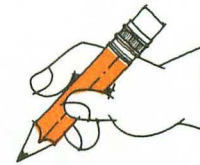
In this mode it is 8 lines high.



GRAPHICS MODE 6

These are the corner coordinates of graphics mode 6. GR.6 has a graphics window of 160 columns by 80 rows. It also has a text window in the last 16 rows of the screen.

The difference between GR.4 and GR.6 is the size of the pixel it can light up. A **pixel** is the smallest block of light a graphics mode can produce.



YOUR TURN

Run this program on the Computer. It will change back and forth between modes 4 and 6.

```
NEW
10 GR.4 : C.1
20 PL.0,0 : PL.79,0
30 PL.0,39 : PL. 79,39
40 F.P=1 TO 500 : N.P
50 GR.6 : C.1
60 PL.0,0 : PL.79,0
70 PL.0,39 : PL. 79,39
80 F.P=1 TO 500 : N.P
90 G. 10
```

Graphics mode 4, color 1.

Light the corner pixels of graphics mode 4.

Pause a second.

Change to mode 6.

Light the same pixels as above.

Pause a second.

Return to mode 4.

Do the lit pixels stay in the same positions in both modes?

Which graphics mode has a smaller pixel, GR.4 or GR.6? Of course, GR.6.

The row and column size tells you how many pixels fit on the screen. The more pixels the screen can contain, the smaller the pixels must be. The smaller pixel can draw a thinner line. So if you are using the Computer to draw lines, GR.4 will quickly cover the screen, while GR.6 can produce finer quality pictures.



CHECKPOINT

1. What graphics command will eliminate the text window in GR.4? _____ GR.6?

2. What are the coordinates for the position at the upper left of the screen? _____
3. In GR.4, along the top edge of the screen $Y =$ _____ and X varies from
_____ to _____. X means columns (across) and Y means rows (down).
4. In GR.6, along the top edge of the screen $Y =$ _____ and X varies from
_____ to _____.
5. In GR.4, at the right edge of the screen $X =$ _____ and Y varies from
_____ to _____.
6. In GR.6, at the right edge of the screen $X =$ _____ and Y varies from
_____ to _____.
7. In GR.4, how many print positions are there along X ? _____
along Y ? _____.
8. In GR.6, how many print positions are there along X ? _____
along Y ? _____.



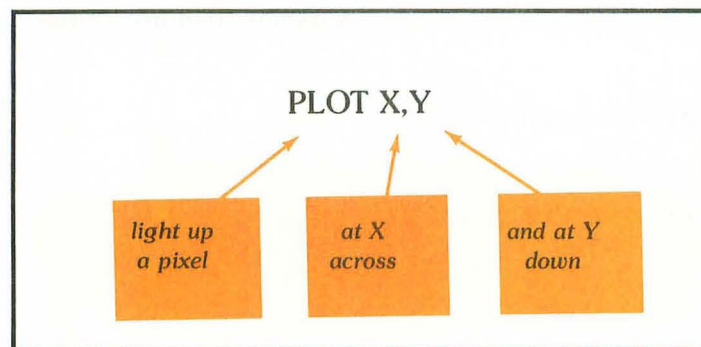
To light up a pixel on the screen two things must happen: the Computer must know what **COLOR** to use and where the pixel is to appear on the screen. The **COLOR** command specifies the color to be used. In the two-color modes **COLOR 1** (or **C.1**) picks the color. Before any drawing is to be done a **COLOR** command must be executed. Then use **PLOT X,Y** (or **PL.X,Y**) to light up a pixel at position X,Y.

Try it.

```
NEW
10 GRAPHICS 4
20 COLOR 1
30 PLOT 60,30
RUN
```

*Use mode 4.
Select color 1 (orange).
Move across 60 and down 30.*

This program lights a pixel at X=60, Y=30.



Let's try this in **GRAPHICS 6** without a text window:

```
NEW
10 GR.6+16 : C.1
20 PL. 60,30
30 G.30
```

*Mode 6 without a text window, COLOR 1.
Plot 60 across, 30 down.
Remain in an infinite loop.*

Remember, with no text window you need to prevent the Computer from printing **READY**. The infinite loop of Line 30 does that.

Now that we can light a square, let's draw a line.

```
NEW
10 GR.6 : C.1

20 PLOT 10,5
30 DRAWTO 150,5
RUN
```

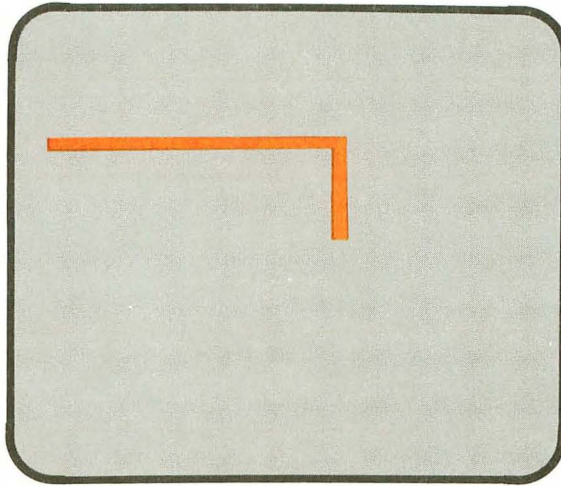
*It is a good idea to put the COLOR and GRAPHICS commands on the same line.
Illuminate location (10,5).
Draw a line to location (150,5).*

The command **DRAWTO** (or **DR.**) will light all the pixels from the **PLOT** location to the position in **DRAWTO**.

There can be as many **DRAWTO**'s in a program as necessary.

```
NEW
10 GR.20 : C.1
20 PL.0,10
30 DR.50,10
40 DR.50,15
50 G.50
RUN
```

GR.4+16 eliminates the text window.



BRAIN FOOD

A DRAWTO always starts drawing from the last lit pixel. A pixel can be lit by PLOT or by DRAWTO.

CHALLENGE

See if you can draw: a. a horizontal line

b. a vertical line

c. a box

d. a triangle

e. a house

LINES GALORE

This unusual program displays many lines on the screen. The length of each line is L. Each line starts on the screen at X=A (column number A) and Y=B (row number B). The length of the line and where it appears on the screen is random. Thus the variables L, A, and B are random.

```

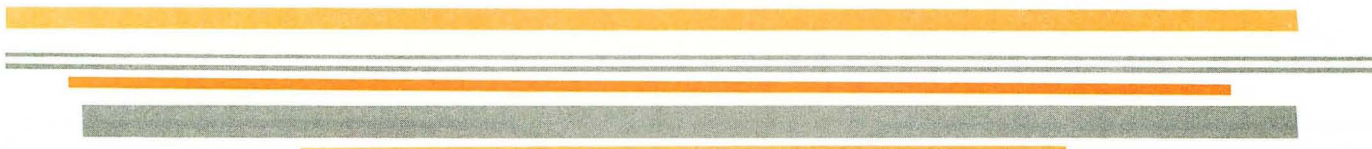
NEW
10 GR.0
20 PR."HOW MANY LINES" : I.N
22 GR.22 : C.1
25 F. K=1 TO N
30 L=INT(160*RND(0))
35 A=INT(160*RND(0))
40 B=INT(96*RND(0))
45 PL. A,B
50 TRAP 60
55 DR. A+L,B
60 N.K
70 G.70
RUN

```

Another way to clear the screen.

Trap errors in case we draw off the screen.

Try it. Wow, that is really unusual. Each time you run the program a new display of lines appears.



THINK

The lines galore program includes random numbers and graphics. Put on your thinking cap and tackle these questions:

1. What is variable L used for? _____

What is variable B used for? _____

2. The length of each graphed line is determined in Line _____

of the program. The longest possible line is _____ graphics blocks long.

3. Each line starts at block position _____ and ends at _____

block position

4. The **TRAP 60** is used to avoid what error?

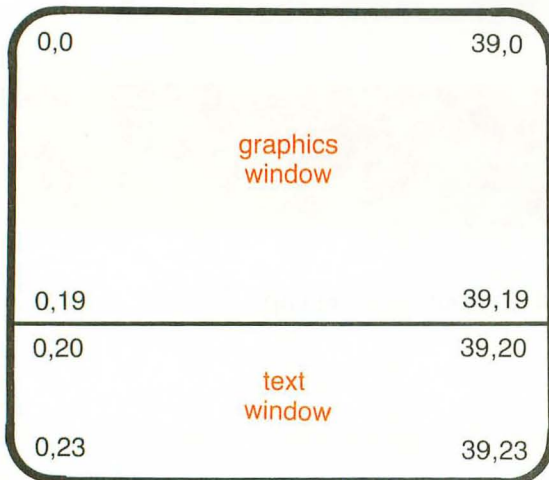
GRAPHICS MODES 3, 5 AND 7

Graphics modes 3, 5, and 7 use four colors. They use one color for the background — **COLOR 0**, and three colors to light the screen's pixels — **C.1**, **C.2**, and **C.3**. Try this.

```
NEW
10 GR.3 : C.1
20 PL.39,19 : DR.19,0
30 C.2 : DR.0,19
40 C.3 : PL.9,10 : DR.30,10
50 END
```

*Select mode 3 and color 1.
Draw a line from (39,19) to (19,0).
Use color 2 to draw a line from (19,0) to (0,19).
Use color 3 to draw a line from (9,10) to (30,10).*

Each time a new **COLOR** is selected, the Computer will draw the line in that color.

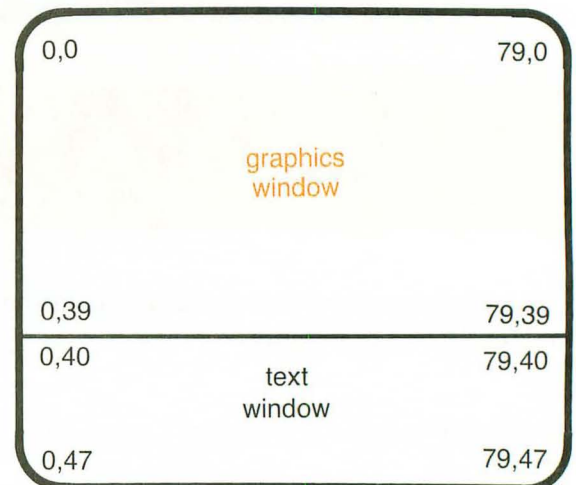


GRAPHICS MODE 3

These are the corner coordinates of graphics mode 3.

40 columns by 20 rows is the size of the graphics window in GR.3.

The text window is 4 lines high.



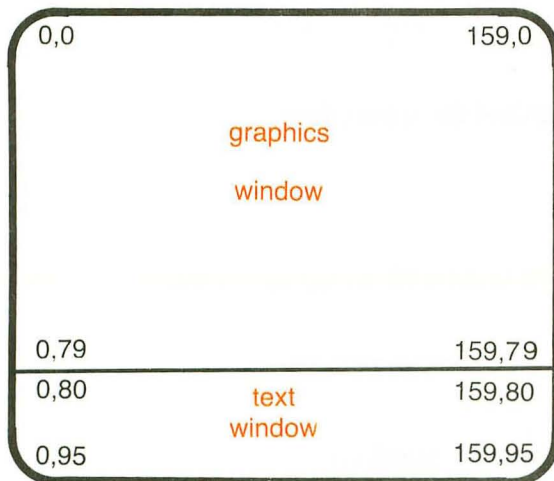
GRAPHICS MODE 5

Graphics mode 5 can light pixels that are smaller.

GR.5 is 80 columns by 40 rows for the graphics window.

It has a text window of 8 rows.

The screen in graphics mode 5 is the same size as the screen of graphics mode 4. **GR.5** and **GR.4** light the same number of pixels.. In **GR.5** the pixels can be in 3 different colors at the same time.



GRAPHICS MODE 7

*Graphics mode 7 has even smaller pixels.
The graphics window is 160 columns by 80 rows.*

The last 16 rows of the screen form the text window.

The screen in graphics mode 7 is the same size as graphics mode 6. **GR.6** lights its pixels in only one color. **GR.7** lights the pixels in 3 different colors at the same time.

EXPERIMENT



PRINT #6; is used in the text modes 1 and 2 to display characters on the screen. Try using **PR.#6;** in graphics modes 3 through 8. Choose a mode, then use these lines in a program:

```
PR.#6;142
PR.#6;"HELLO"
PR.#6;"good-bye"
```

Change the characters to inverse video. Notice that the characters use all four color registers. The register used depends on the code for the character being printed.

BRAIN FOOD

There are 2 ways to clear the screen in a program:

1. PRINT "␣"
2. The GRAPHICS command for example:

```
GR. 1
```

TURN IT OFF

COLOR together with **PLOT X,Y** or **DRAWTO X,Y** turns on graphics blocks. How are they turned off?

The **GRAPHICS** command clears the entire screen.

If only certain lines are to be erased, draw over them using the background color, **COLOR 0**.

```
NEW
```

```
10 GR.7
```

```
15 C.2
```

```
20 FOR X=10 TO 30 STEP 3
```

```
30 PL.X,0 : DR.X,79
```

```
40 NEXT X
```

```
50 C.0
```

```
60 FOR X=10 TO 30 STEP 3
```

```
70 PL.X,0 : DR.X,79
```

```
80 NEXT X
```

```
90 GOTO 15
```

The screen is cleared.

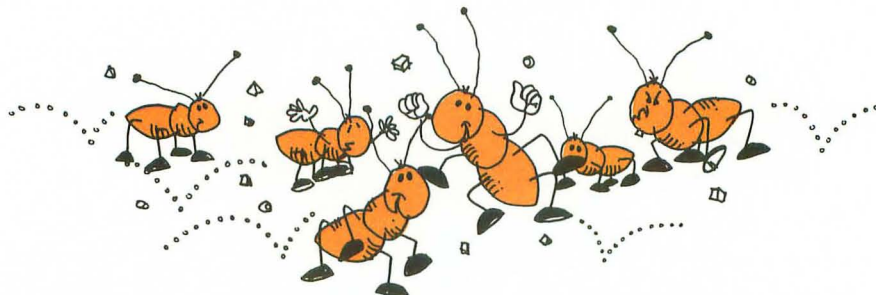
Select COLOR 2 to draw.

Now you see it.

Select the background color to erase.

Now you don't see it.

Run the program. Have you seen enough? To stop the display press the **BREAK** key. Otherwise, the display will go on forever.



TERMITE TIME

Here come the termites. This program demonstrates what could happen when termites invade the Computer's screen.

NEW

5 REM LIGHT UP THE SCREEN

10 GR. 3+16

15 FOR X=0 TO 39

20 C=INT(3*RND(0))+1 : C.C

Choose a random color.

25 PL. X,0 : DR. X,23

30 NEXT X

40 REM LET THE TERMITES EAT AWAY

45 C.0

Use the background color.

50 X=INT(40*RND(0))

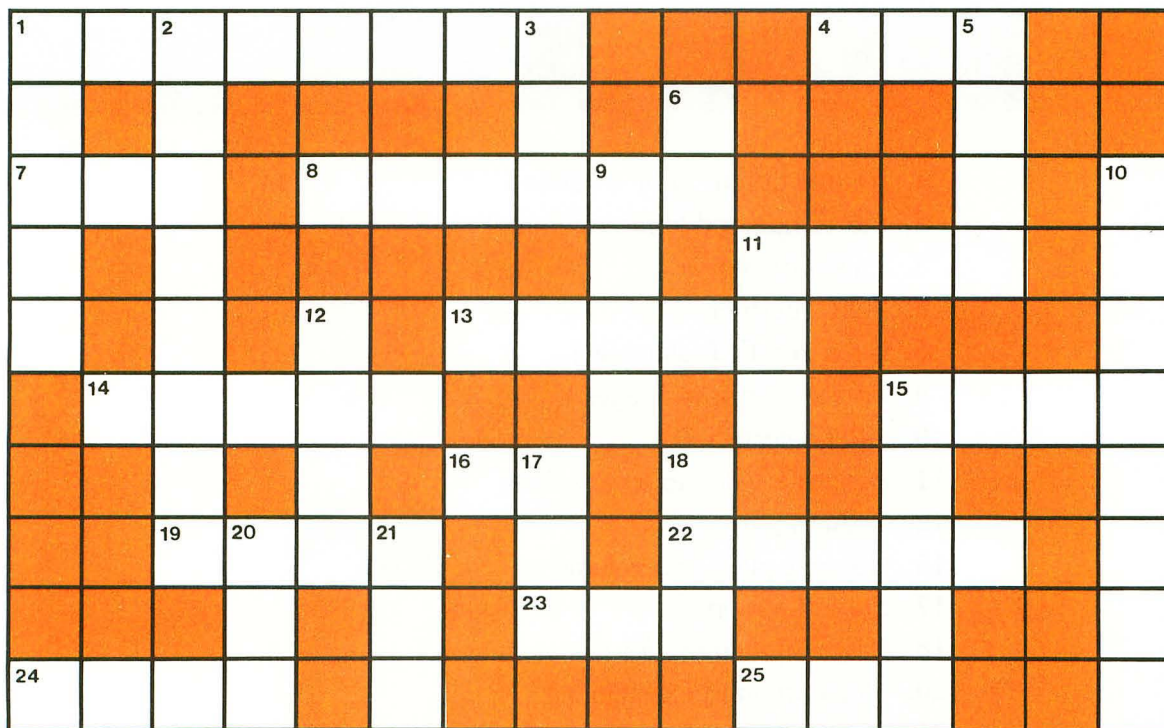
60 Y=INT(24*RND(0))

70 PL. X,Y

80 GOTO 50

Run the program and watch the screen light up completely. Then the termites eat away at the screen. The graphics blocks are turned off randomly by coloring them black.

FUN TIME



Fill in this crossword puzzle.

ACROSS

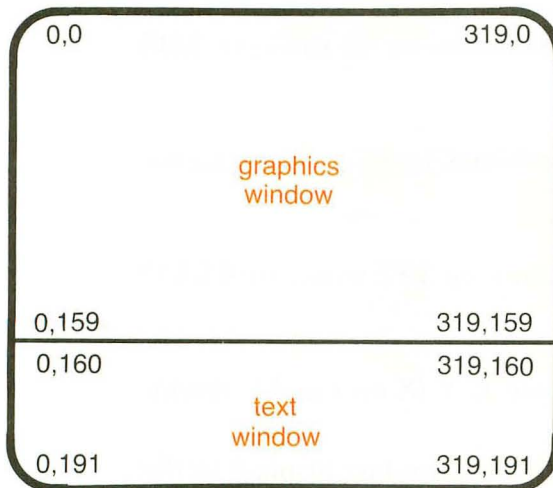
1. Command to move the cursor.
4. Tells the Computer to execute the program.
7. Drops the numbers after the decimal point.
8. Displays a line from the last cursor position.
11. Display the program on the screen.
13. Copy the program onto a cassette.
14. Get a program from a cassette.
15. It increments a FOR-NEXT loop.
16. A question with a condition.
19. DATA's partner.
22. Tells the Computer to ask for data with a ?.
23. Allows the programmer to comment.
24. Input from disk to Computer.
25. Chooses a surprise number from 0 to 1.

DOWN

1. Causes output to be displayed.
2. Stores hue and luminance numbers in a color register.
3. Erases memory.
5. FOR's partner.
6. Used in a FOR statement.
9. Prevents error messages.
10. GR.1 selects a _____ mode.
11. Assigns a value to a variable.
12. READ's partner.
15. Controls pitch and volume of note played.
17. Start of a loop.
18. Reserves memory for strings.
20. Stops program execution.
21. Disk Operating System.

GRAPHICS MODE 8

Mode 8 offers the best quality line in ATARI Home Computer graphics. In this mode the screen contains over 61,000 pixels.



GRAPHICS MODE 8

The full screen has 192 rows with 320 pixels per row. This is a total of 61,440 pixels ($192 \times 320 = 61,440$).

Only one color is possible at one time on the screen. It is called by **COLOR 1** and has a separate brightness for the graphics point.

Try this program to see the color of the background and the color of the pixel. When the program asks "how bright" be careful you don't use the number 10. 10 is the brightness value of the pixel itself.

```

NEW
5  REM draw a letter M
10 GR.8 : C.1
20 PL.5,191 : DR.5,5
30 DR.160,191 : DR.315,5
40 DR.315,191
45 REM change the background color
50 PR."Which color (0 - 15)";
55 I. COLOR
60 PR."How bright (0 - 14)";
65 I. LIGHT
70 SE.2,COLOR,LIGHT
80 G.50
  
```

Select mode 8.
Draw a large M in the graphics window.

Request the background color for Line 70.

Request the brightness of the background for Line 70.

Change the background.
Return to Line 50.



TIME OUT FOR OLD NEWS

- Graphics mode 3 has 40 columns by 24 rows, or 960 pixels.
- Graphics modes 4 & 5 have 80 columns by 48 rows, or 3840 pixels.
- Graphics modes 6 & 7 have 160 columns by 96 rows, or 15,360 pixels.
- Graphics mode 8 has 320 columns by 192 rows, or 61,440 pixels.
- PLOT X,Y lights a pixel at coordinate X, Y (X over and Y down).
- DRAWTO X,Y lights all the pixels from the last lit pixel to the pixel at coordinate X,Y.

EXERCISES

1. Which graphics mode draws in 2 colors and large blocks?

2. Which graphics mode draws in 4 colors and small blocks?

3. How can you eliminate the text window in GR.7?
What possible problem can arise?

4. Write a program to draw an orange box.

5. Change the program in #4 to draw a solid orange box.

6. Write a program in GR.7 to draw a star and display "twinkle, twinkle".

7. What is the output of each of the following programs:

a. NEW

```
10 GR.8 : C.1
20 PL. 50,50 : DR. 50,100
30 PL. 50,75 : DR. 100,75
40 PL. 125,50 : DR. 175,50
50 PL. 150,50 : DR. 150,100
60 PL. 125,100 : DR. 175,100
70 PL. 100,50 : DR. 100,100
80 END
```

b. NEW

```
10 GR. 5 : C.2
20 PL. 20,1 : DR. 20,9
30 C. 3 : PL. 0,0 : DR. 79,0
40 C. 3 : PL. 0,22 : DR. 79,22
50 C. 1 : PL. 79,0 : DR. 79,22
60 C. 1 : PL. 0,0 : DR. 0,22
70 C. 2 : PL. 1,9 : DR. 20,9
```

c. NEW

```
10 GR. 7 : C. 1
20 X=INT(60*RND(0)+100)
30 FOR Y=0 TO 95
40 C. 1 : PL. 0,Y : DR. X,Y
50 C. 0 : PL. 100,Y
60 DR. X,Y
70 NEXT Y
80 END
```

d. NEW

```
10 GR. 5+16 : C. 1
20 PL. 0,0 : DR.0,47
30 DR. 79,47 : DR. 79,0 : DR. 0,0
40 C. 2
50 PL. 20,12 : DR. 20,36
55 DR. 60,36
60 DR. 60,12 : DR. 20,12
70 G. 70
```

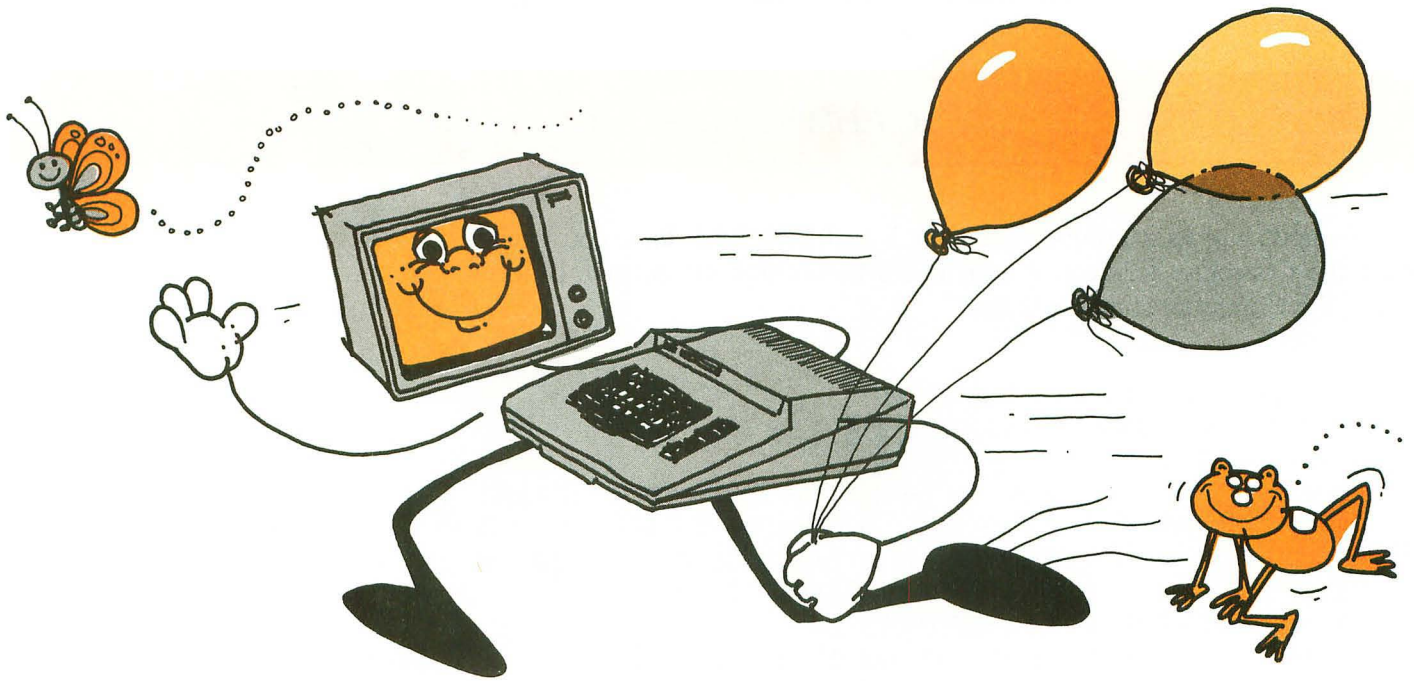
8. Write a program to:

a. Display the word GOODY, in letters as tall as the screen.

b. Draw three nested triangles (the triangles are inside each other) in different colors.

b. Draw three nested triangles (the triangles are inside each other) in three different colors.

A large rectangular area with horizontal yellow lines, intended for drawing three nested triangles in different colors. The lines are evenly spaced and extend across the width of the drawing area.



CHAPTER 14

PROGRAMS TO GO

Sometimes it is fun to just copy a program and type it in. Here are some programs for your collection.

1. HI-LOW WITH A LYING COMPUTER
2. TIMED ARITHMETIC DRILL
3. MASTERFIND
4. SHOOT THE DUCK
5. FIREWORKS
6. ETCH-A-SKETCH
7. WHEEL OF FORTUNE

The programs use some BASIC statements which you have not learned. You don't have to understand the programming, but you must type in the programs exactly as shown. Don't forget to save them on a cassette or on disk. Have fun!

1. HI-LOW WITH A LYING COMPUTER

The Computer picks a number between 1 and 100. Each time you enter a guess the Computer tells you if your guess is too high or too low. Unfortunately, the Computer sometimes lies!

```

HI-LOW WITH A LYING COMPUTER
1  REM HI-LOW WITH A LYING COMPUTER
5  PRINT CHR$(125)
10 PRINT "      HI-LOW WITH LYING COMPUTER":PRINT :PRINT :GOSUB 1000
30 PRINT "I AM THINKING OF A NUMBER BETWEEN 1   AND 100.":PRINT
40 PRINT "I WILL TELL YOU IF YOUR GUESS IS TOO  HIGH OR TOO LOW, BUT ...
   BEWARE.....      SOMETIMES I LIE!!!"
50 N=INT(100*RND(0))+1:PRINT
60 PRINT "HOW OFTEN DO YOU WANT ME TO LIE?      OFTEN=1, SOMETIMES=2,
   SELDOM=3"
70 INPUT X:X=X+2:PRINT CHR$(125):K=0
80 K=K+1:PRINT "THIS IS GUESS NUMBER ":K:PRINT "YOUR GUESS IS":INPUT
   G:PRINT
90 IF N>G THEN 120
100 IF N<G THEN 150
110 PRINT "YOU DID IT IN ";K;" GUESSES.":END
120 IF INT(X*RND(0))+1<X THEN 140
130 PRINT "YOUR GUESS IS TOO HIGH - TRY AGAIN":GOTO 80
140 PRINT "YOUR GUESS IS TOO LOW - TRY AGAIN":GOTO 80
150 IF INT(X*RND(0))+1<X THEN 130
160 GOTO 140
1000 FOR ZA=1 TO 100:NEXT ZA:RETURN

```

2. TIMED ARITHMETIC DRILL

Check how long it takes you to do 5 problems. Select among addition, subtraction, multiplication and division. You may also select how difficult the questions will be. For each problem you get wrong on the first try, you lose 5 seconds and for each problem you get wrong on the second try, you lose 10 more seconds.

Compete with your friends and become a math whiz!

```

1  REM TIMED ARITHMETIC DRILL
5  DIM M$(4),Q$(1),N$(1),Z$(4)
10 REM ARITHMETIC
20 PRINT CHR$(125)
30 PRINT "THIS IS A TIMED DRILL FOR FIVE      ARITHMETIC PROBLEMS.":PRINT
40 PRINT "THE TOTAL ELAPSED TIME FROM THE START COUNTS.":PRINT
50 PRINT "ALSO, FOR EVERY PROBLEM YOU GET WRONG ON THE FIRST ";
60 PRINT "TRY YOU ARE PENALIZED      FIVE SECONDS.":PRINT
70 PRINT "FOR EVERY PROBLEM YOU GET WRONG ON THESECOND TRY ";
80 PRINT "YOU ARE PENALIZED TEN      SECONDS.":PRINT

```



```

90 PRINT :Z$="+-*/"
100 SETCOLOR 2,8,4:PRINT "WHICH OPERATION WOULD YOU LIKE      (+,-,* OR
    /)":INPUT M$
110 FOR X=1 TO 4:IF Z$(X,X)=M$ THEN 130
120 NEXT X:GOTO 100
130 PRINT "WHAT LEVEL WOULD YOU LIKE(1,2,3)":INPUT Y
140 IF Y>0 AND Y<4 THEN 160
150 GOTO 130
160 TRAP 280:RESTORE :FOR Q=1 TO 4:FOR R=1 TO 3:READ A1,A2,B1,B2,D1
170 IF Q=X AND R=Y THEN 190
180 NEXT R:NEXT Q
190 PRINT CHR$(125):POSITION 2,10:PRINT "TIMER STARTS WHEN YOU PRESS
    RETURN"
200 INPUT Q$:POKE 19,0:PRINT CHR$(125):H=0
205 T=0
210 H=H+1:IF H>5 THEN 230:GRAPHICS 0:SETCOLOR 2,H,6
215 PRINT CHR$(125):PRINT "PROBLEM # ";H
220 ON X GOTO 300,350,400,450
230 GRAPHICS 0:SETCOLOR 2,1,6:PRINT :PRINT " YOUR ELAPSED TIME IS ";60/16
    *PEEK(19);" SECONDS":PRINT
240 PRINT "YOUR PENALTY FOR WRONG ANSWERS ON THE FIRST TRY IS ";FIRST*5;"
    SECONDS":PRINT
250 PRINT "YOUR PENALTY FOR WRONG ANSWERS ON THE SECOND TRY IS ";SEC*10;"
    SECONDS":PRINT
260 PRINT " YOUR TOTAL TIME IS ";60/16*PEEK(19)+5*FIRST+10*SEC;" SECONDS"
    :PRINT :PRINT
270 PRINT "DO YOU WANT TO GO AGAIN(Y OR N)":INPUT N$:IF N$="Y" THEN PRINT
    CHR$(125):GOTO 100
280 END
300 GOSUB 500:IF A+B>D THEN 300
310 E=A+B:GOTO 600
350 GOSUB 500:IF A-B<D THEN 350
360 E=A-B:GOTO 600
400 GOSUB 500:E=A*B:GOTO 600
450 GOSUB 500:E=A/B:GOTO 600
500 A=INT(A1*RND(0))+A2:B=INT(B1*RND(0))+B2:D=INT(D1*RND(0))
510 RETURN
520 DATA 8,1,9,1,19,79,10,75,10,99,799,100,799,100,999
530 DATA 18,1,18,1,0,78,21,78,11,10,799,222,799,100,100
540 DATA 8,1,9,0,0,9,10,7,2,0,99,100,7,2,0
550 DATA 8,1,8,1,0,9,10,7,2,0,4,15,9,10,0
600 T=T+1:IF T>2 THEN 700
610 POSITION 1,5:PRINT A;M$;B:"=":POSITION 19,5:INPUT V
620 IF V=E THEN POSITION 8,10:T=0:SETCOLOR 2,3,14:PRINT "CORRECT":GOSUB
    730:GOTO 210
630 SETCOLOR 2,6,2:POSITION 8,9:PRINT "WRONG":GOSUB 730:IF T>1 THEN 700
640 FIRST=FIRST+1:POSITION 20,5:PRINT "      ":GOTO 600
700 POSITION 10,10:PRINT " THE CORRECT ANSWER IS ";E
710 GOSUB 730:SEC=SEC+1:GOTO 205
720 GOSUB 730:GOTO 210
730 FOR V=1 TO 500:NEXT V:RETURN

```


3. MASTERFIND

A game similar to Mastermind. The Computer selects a number and you try to figure out each digit in as few guesses as possible.

```

5 DIM R$(5),G$(5),Z$(1),R(5),S$(1)
6 GRAPHICS 0:SETCOLOR 2,13,0
10 PRINT "MASTERFIND":PRINT
20 PRINT " THE ATARI PICKS A RANDOM NUMBER OF      FIVE DIGITS. YOU TRY TO
   GUESS IT."
30 PRINT :PRINT "THE COMPUTER RESPONDS WITH THE NUMBER OF POSITIONS(P'S)
   AND THE NUMBER OF      DIGIT
40 PRINT :PRINT "THE P'S MEAN CORRECT DIGITS CORRECTLY PLACED."
50 PRINT "THE D'S MEAN CORRECT DIGITS, BUT      INCORRECTLY PLACED."
55 FOR Q=1 TO 2000:NEXT Q
60 PRINT :PRINT "SUPPOSE THE ATARI PICKS ----19506"
70 PRINT "=====
80 PRINT "YOUR GUESS IS 15378 THEN P=1 AND D=1."
90 PRINT "YOUR GUESS IS 19078 THEN P=2 AND D=1."
100 PRINT "YOUR GUESS IS 19578 THEN P=3 AND D=0.":PRINT
103 PRINT "IF YOU WANT TO END THE GAME, PRESS      RETURN WITHOUT A NUMBER.
   THE COMPUTER WILL TELL YOU IT'S NUMBER"
105 PRINT :PRINT "PRESS RETURN TO START - READY":INPUT S$
110 PRINT CHR$(125):R(1)=INT(10*RND(0))
115 FOR A=2 TO 5
120 R(A)=INT(10*RND(0)):FOR B=1 TO A-1
130 IF R(A)=R(B) THEN 120
140 NEXT B
150 NEXT A
160 FOR A=1 TO 5:R$(A,A)=STR$(R(A)):NEXT A
165 K=1:SETCOLOR 2,1,4:SETCOLOR 1,1,14
170 PRINT "THIS IS GUESS NUMBER ";K;" ENTER A FIVE      DIGIT NUMBER"
180 INPUT G$
185 IF LEN(G$)=0 THEN 305
190 IF LEN(G$)=5 THEN 210
200 PRINT "FIVE DIGITS PLEASE.":GOTO 180
210 D=0:P=0:FOR A=1 TO 5:FOR B=1 TO 5
220 IF R$(A,A)=G$(B,B) THEN D=D+1
230 NEXT B:NEXT A
240 FOR A=1 TO 5
250 IF R$(A,A)<>G$(A,A) THEN 270
260 P=P+1:D=D-1
270 NEXT A
280 IF P=5 THEN 300
290 PRINT "P = ";P;" D = ";D:PRINT :K=K+1:GOTO 170
300 GRAPHICS 0:SETCOLOR 2,10,4:POSITION 5,10:PRINT "YOU DID IT IN ";K;"
   GUESSES."
304 FOR ZQ=1 TO 5:PRINT CHR$(253);:NEXT ZQ:GOTO 310
305 PRINT "THE COMPUTER'S NUMBER IS ";R$:FOR Z2=1 TO 1500:NEXT Z2
310 PRINT CHR$(125):PRINT " DO YOU WANT TO PLAY AGAIN (Y OR N)":INPUT Z$
320 IF Z$="Y" THEN 110
330 END

```

4. SHOOT THE DUCK

Do you like Arcade games? Well here is a simple version of a shooting gallery. A duck moves back and forth across the screen. You have 60 seconds to hit it as often as you can. Your score is displayed as the number of hits and the percent accuracy.

```

1 REM SHOOT THE DUCK
5 DIM S$(1),D$(2),U$(2),K$(3),G$(1),A$(1),TITLE$(20)
6 F=0:TITLE$="SHOOT THE DUCK"
8 POKE 19,0:POKE 752,1
10 GOSUB 3000:GRAPHICS 0:POSITION 2,10:POKE 752,1
20 PRINT "UP ARROW FIRES THE GUN,> MOVES THE GUN TO THE RIGHT"
30 PRINT "< MOVES THE GUN TO THE LEFT. YOU HAVE 60 SECONDS TO PLAY."
40 PRINT "HAVE FUN!":PRINT "PRESS RETURN TO START.":INPUT S$
50 D$(1,1)=CHR$(9):D$(2,2)=CHR$(15)
60 U$(1,1)=CHR$(143):U$(2,2)=CHR$(25)
70 K$(1,1)=" ":K$(2,2)=CHR$(160):K$(3,3)=CHR$(160)
80 G$=CHR$(160)
85 X=20:G=20:GOSUB 1000
86 IF 60/16*PEEK(19)=60 THEN 800
90 IF FLAG=1 THEN 110
100 X=X+1:IF X>37 THEN X=37:FLAG=1
103 GOSUB 1000
105 GOTO 120
110 X=X-1:IF X<0 THEN X=0:FLAG=0
115 GOSUB 1000
120 IF PEEK(764)=55 THEN G=G+2:POKE 764,255
130 IF PEEK(764)=54 THEN G=G-2:POKE 764,255
140 IF PEEK(764)=14 THEN POKE 764,255:GOTO 200
150 GO TO 86
200 F=F+1:FOR D=X TO X+3:FOR QQ=20 TO 5 STEP -5:POSITION G,QQ:PRINT CHR$(124);NEXT QQ:IF G=D THEN 210
205 NEXT D:GOTO 86
210 PRINT "QUACK":POSITION X,3:PRINT "  ":POSITION X,4:PRINT "  ":POSITION X,5:PRINT "  ":H=H+1:GOSUB 2000
215 GOTO 86
300 PRINT " TIME IS UP. YOU FIRED ";F;" SHOTS AND HIT ";H;" TIMES. YOUR ACCURACY IS ";100*H/F;" %."
310 PRINT "DO YOU WANT TO PLAY AGAIN (Y OR N)":INPUT A$:IF A$="Y" THEN 85
315 END
1000 PRINT CHR$(125):POSITION X,3:PRINT D$:POSITION X,4:PRINT U$:POSITION X,5:PRINT K$
1001 IF G>39 THEN G=39
1002 IF G<0 THEN G=0
1003 POSITION G,20:PRINT G$:RETURN
2000 FOR PAUSE=1 TO 200:NEXT PAUSE:RETURN
3000 GRAPHICS 19:FOR COLOR=1 TO 15
3010 POSITION 0,5*(20-LEN(TITLE$)),6:PRINT #6:TITLE$
3020 SETCOLOR 0,COLOR,6:FOR PAUSE=1 TO 100:NEXT PAUSE:NEXT COLOR
3030 RETURN

```




5. FIREWORKS

Here are explosions of sound and color. This program uses the joystick in controller jack #1. Just press the button and an explosion appears on the screen. Can you fill the screen? Watch the pretty patterns form and listen to the noise. Have fun!

```

1 REM FIREWORKS
10 GRAPHICS 3+16:SETCOLOR 4,0,0
20 A=INT(RND(0)*16):B=RND(0)*11+4:C=INT(RND(0)*3)+1
30 X=RND(0)*39+1:Y=RND(0)*23+1
40 T=RND(0)*213+29:U=RND(0)*213+29:Q=(T-U)/10
50 SOUND 0,0,0,0:SETCOLOR C-1,A,B:TRAP 20:COLOR C
60 IF STRIG(0)<>0 THEN 60
70 PLOT X,Y:SOUND 0,T,10,6
80 PLOT X+1,Y:PLOT X-1,Y:PLOT X,Y-1:PLOT X,Y+1:L=80:SOUND 0,T+Q,10,6
90 PLOT X+2,Y:PLOT X-2,Y:PLOT X,Y-2:PLOT X,Y+2:L=90:SOUND 0,T+2*Q,10,6
100 PLOT X+1,Y+1:PLOT X-1,Y-1:PLOT X+1,Y-1:PLOT X-1,Y+1:L=100:SOUND 0,T+3
    *Q,10,6
110 PLOT X+3,Y:PLOT X-3,Y:PLOT X,Y-3:PLOT X,Y+3:L=110:SOUND 0,T+4*Q,10,6
120 PLOT X-1,Y-2:PLOT X+1,Y+2:PLOT X-2,Y+1:PLOT X+2,Y-1:L=120:SOUND 0,T+5
    *Q,10,6
130 PLOT X-2,Y-1:PLOT X+2,Y+1:PLOT X-1,Y+2:PLOT X+1,Y-2:L=130:SOUND 0,T+6
    *Q,10,6
140 PLOT X-1,Y-3:PLOT X+1,Y+3:PLOT X-2,Y-2:PLOT X+2,Y+2:L=140:SOUND 0,T+7
    *Q,10,6
150 PLOT X-1,Y+3:PLOT X+1,Y-3:PLOT X-2,Y+2:PLOT X+2,Y-2:L=150:SOUND 0,T+8
    *Q,10,6
160 PLOT X-3,Y-1:PLOT X+3,Y+1:PLOT X-3,Y+1:PLOT X+3,Y-1:L=160:SOUND 0,T+9
    *Q,10,6
170 PLOT X-4,Y:PLOT X+4,Y:PLOT X,Y-4:PLOT X,Y+4:L=170:SOUND 0,T+10*Q,10,6
180 GOTO 20
200 IF X>40 THEN X=X-40:GOTO L+10
210 IF X<0 THEN X=40+X:GOTO L+10
220 IF Y>20 THEN Y=Y-20:GOTO L+10
230 IF Y<0 THEN Y=20+Y:GOTO L+10
240 GOTO 20

```

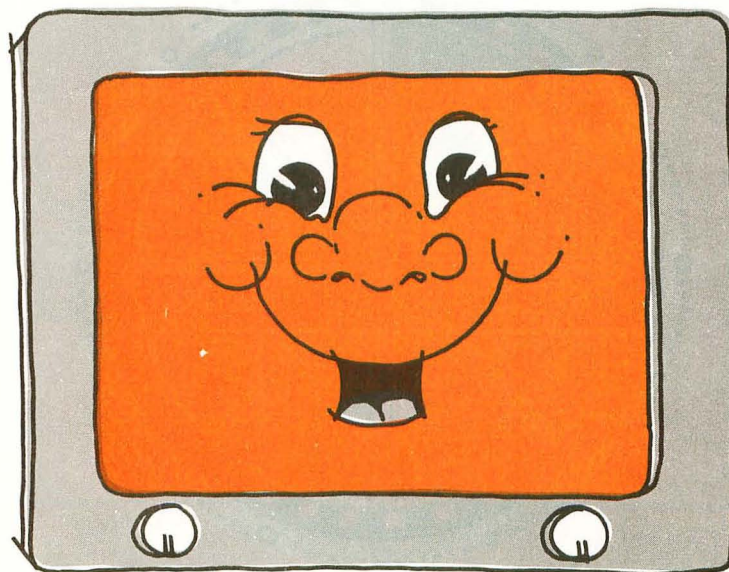

6. ETCH-A-SKETCH

This program uses the joystick in controller jack #1. The program works just like the well known toy. You can draw and erase horizontal, vertical, and diagonal lines. Use the program to write messages or draw pictures. Press the button and you will draw in a different color. How artistic can you be? It's up to you. Try it, it's fun!

```

1 REM ETCH-A-SKETCH
10 GRAPHICS 3+16:SETCOLOR 4,6,2:X=20:Y=12:C=0
15 SETCOLOR 0,8,10:COLOR 1:PLOT X,Y
20 A=RND(0)*16:B=RND(0)*11+4:C=C+1
30 IF C>3 THEN SETCOLOR 4,A,2:C=0:GOTO 50
40 SETCOLOR C-1,A,B:COLOR C:Q=C
50 TRAP 170
55 FOR P=1 TO 100:NEXT P
65 IF STRIG(0)=0 THEN 20
70 ON STICK(0)+4 GOTO 90,100,110,60,120,130,140,60,150,160
80 GOTO 60
90 X=X+1:Y=Y+1:GOTO 220
100 X=X+1:Y=Y-1:GOTO 220
110 X=X+1:GOTO 220
120 X=X-1:Y=Y+1:GOTO 220
130 X=X-1:Y=Y-1:GOTO 220
140 X=X-1:GOTO 220
150 Y=Y+1:GOTO 220
160 Y=Y-1:GOTO 220
170 IF X>40 THEN X=0:GOTO 50
180 IF X<0 THEN X=40:GOTO 50
190 IF Y>24 THEN Y=0:GOTO 50
200 IF Y<0 THEN Y=24:GOTO 50
210 GOTO 50
220 LOCATE X,Y,Z
230 IF Q=Z THEN COLOR 0:PLOT X,Y:COLOR C:GOTO 60
240 PLOT X,Y:GOTO 60

```



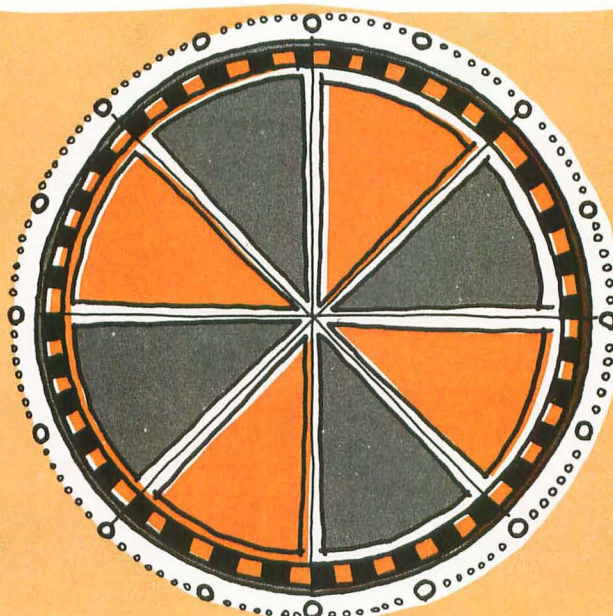
7. WHEEL OF FORTUNE

Have you ever been to a carnival or an amusement park and watched the Wheel Of Fortune? These are the games where you bet on the spin of a wheel. This can get very expensive! The program is just like the carnival game, but it's free! Type it in and see how lucky you are. Try competing with a friend. You're sure to have fun.

```

1 REM WHEEL OF FORTUNE
2 DIM A$(1),G$(1):OPEN #1,4,0,"K:"
5 PRINT CHR$(125):PRINT "THE WHEEL OF FORTUNE WILL SPIN THE"
6 PRINT "          NUMBERS FROM 1 TO 6."
7 FOR Q=1 TO 1500:NEXT Q
10 GRAPHICS 1+16:POSITION 0,9
20 PRINT #6;"YOUR LUCKY NUMBER IS":GET #1,N:G$=CHR$(N)
30 POSITION 10,11:PUT #6,N
35 L=INT(6*RND(0)+1):K=0
40 FOR A=1 TO 50:SOUND 0,60,14,10
50 POSITION 10,0:PRINT #6;INT(6*RND(0)+1)
55 FOR W=1 TO 5:NEXT W:SOUND 0,0,0,0
60 FOR W=1 TO A/15*10:NEXT W:POSITION 10,0:PRINT #6;" ":NEXT A
65 POSITION 10,0:PRINT #6;L:FOR W=1 TO 100:NEXT W
70 IF L=VAL(G$) THEN 95
80 GRAPHICS 0:POSITION 5,12:PRINT "THE LUCKY NUMBER IS ":L
90 POSITION 15,14:PRINT "YOU DID NOT HAVE IT":GOTO 110
95 GRAPHICS 2:SETCOLOR 4,4,4:SETCOLOR 2,4,4
100 PRINT "THE LUCKY NUMBER IS ":L
105 POSITION 4,2:PRINT #6;"YOU GOT IT!":POSITION 4,5:PRINT #6;"you got
    it!"
110 POSITION 2,23:PRINT "TO PLAY AGAIN PRESS THE A KEY";:GET #1,N
120 IF CHR$(N)="A" THEN 10
130 END

```



ANSWERS

CHAPTER 1

CHECKPOINT:

1. So the zero looks different from the letter O.
2. Yes all the letter keys are upper case.
3. All the keys containing symbols other than letters and digits are special keys.

EXPERIMENT:

The screen is completely erased.

CHALLENGE:

38 characters in each line, 24 lines on the screen.

CHAPTER 2

YOUR TURN:

15 9 36 4

THINK:

1. + addition - subtraction * multiplication /division

EXPERIMENT

1. The space between **PRINT** and the quotation marks is not necessary.
2. Messages and calculations can be combined. The semicolon is used between them.
3. A blank line is displayed because a blank message is within the quotation marks.

SHORTCUT:

They all display 36500.

CHALLENGE:

"5 + 3" is a message and not an addition.

FUN TIME:

¹	R	E	T	U	R	N		
	E							
	A				² S			³ Q
⁴	D	E	L	E	T	E		U
	Y				A			O
				⁵ P	R	I	N	T
								E
			⁶ B	A	C	K	S	

EXERCISES:

1. AROUND THE WORLD IN 80 DAYS
2. PRINT 7*24*60
3. PRINT "SIX FEET EQUAL ";6*12*2.54;" CENTIMETERS."
4. PRINT "MR. PRESIDENT"
PRINT "1600 PENNSYLVANIA AVENUE"

CHAPTER 3

YOUR TURN:

The background is dark purple.

The background is bright pink.

The border is dark green.

The border is mid orange-green.

YOUR TURN:

A low tone, soft, pure sound.

A middle tone, medium volume, distorted sound.

A very high tone, slightly soft, throbbing sound.

A very high tone, slightly loud, click.

Sound register 0 is turned off.

A middle tone, slightly soft, pure sound.

FUN TIME:

1. SOUND
2. CTRL
3. SETCOLOR
4. DELETE
5. CLEAR
6. SYSTEM RESET

D	E	L	A	S	C	M	Q	R
N	S	O	B	R	L	E	I	O
S	O	E	A	R	E	A	D	S
S	U	N	T	N	A	M	E	D
O	N	C	E	C	R	A	K	L
N	D	U	C	L	O	R	E	A
D	E	L	E	T	E	L	T	R
N	C	O	L	T	C	U	O	S
U	S	Y	S	T	E	M	L	R
O	C	R	E	S	E	T	O	C

EXERCISES:

1. Editing
2. **CTRL** **DELETE**
3. **SHIFT** **INSERT**
4. Use the **CAPS LOWR** key.
5. 16 (8 shades of each color)
6. **SOUND & SETCOLOR**
7. For example, **SE. 2,6,4** and **SO.1,50,10,8** are nice.

CHAPTER 4

CHECKPOINT:

1. 10 LET L = 20
2. Line 40
3. The program is gone from memory.

THINK:

- a) I AM GREAT
- b) 132.85 63.79 3394.9896 2.8473909
- c) 5678+1234 EQUALS 6912
- d) 10 100
 20 400
 30 900

YOUR TURN:

```
A=1
B=2
PRINT A
1
PRINT B
2
PRINT E
0
PRINT A,B,E
1              2              0
```

EXERCISES:

1. **PRINT LET**
2. **6+10 = 16**

3. **ERROR** -; A+3 is an illegal variable name; therefore a syntax error occurs.

4. 111 11

5. HELLO

96
2300
4

6. NEW

10 EGG = 1.4
20 NUMBER = 12
30 PRINT EGG*NUMBER
40 END

7. Move the cursor to the number 12 of Line 20. Type **50** and press **RETURN**.

CHAPTER 5

LET'S PRETEND

14 -2 48 .75
READY

2*M+3*R=36
READY

A DISCOVERY:

$94+89+90/3 = 213$; only the 90 is divided by 3.

The double commas are needed to line up the first two lines of answers with the third line. The statement "AVERAGE GRADE" is longer than 10 characters, therefore we use more than 1 zone.

FUNTIME:

B	C	D	A	T	A	O
Z	A	T	I	N	E	L
C	Q	S	W	S	Q	A
S	K	F	I	E	I	U
T	E	L	T	C	N	Q
T	N	I	R	P	V	E
B	K	R	D	N	T	A
W	N	T	A	F	S	M
L	U	M	E	E	I	M
N	R	J	R	V	L	O
C	R	E	A	D	Y	C

EXERCISES:

1. A A\$

2. `Ø`
blank string
3.
 - a. Remove the comma before A.
 - b. 13 should be a variable name.
 - c. X-Y is not a legal variable name.
 - d. Only one comma between 5 and 6.
 - e. No error
 - f. Period should be a comma or a semicolon.
 - g. Missing a value for C.
4.
 - a. JACK AND JILL
 - b. 6 `Ø` 13 `Ø`
5. NEW

```
1Ø READ A,B,C
15 DATA 2,3,4
2Ø D = A*B-C
25 PRINT D
3Ø END
```
6. NEW

```
1Ø PR. "    X"
2Ø PR. "   XXX"
3Ø PR. "  XXXXX"
4Ø PR. " XXXXXXX"
5Ø PR. "   XXX"
6Ø PR. "   XXX"
7Ø PR. "   XXX"
```

CHAPTER 6

CHECKPOINT:

1. CSAVE
2. together press the **RECORD** and **PLAY** buttons.
3. CLOAD: **RETURN** key; **PLAY** button; **RETURN** key.

CHALLENGE:

change or insert the following lines:

```
24 K=Ø
26 K=K+1
4Ø PR. "***** YOU GOT IT IN ";K;" GUESSES *****"
```

YOUR TURN:

To save the program:

1. Type in the program.
2. Type: **S(A)V(E) (D)1;(C)O(L)O(R)**.

3. Press **RETURN**.

To run the program:

1. Turn on the disk drive, T.V., and Computer in that order.
2. Type: **LOAD "D1:COLOR"**.
3. Press **RETURN**.
4. Type **RUN**.

EXERCISES:

1. Once the program is on tape or disk you can load it into the Computer whenever you want.
2. **CSAVE** stores a program on cassette tape.
3. **SAVE "D1: name"** stores a program on disk. The name can be up to 8 letters or digits.
4. **CLOAD** will load into the Computer whichever program is next on the tape.
5. Type **LOAD "D1:TWO"** and press **RETURN**.
6. **CLOAD** will load a program from a cassette into the Computer.
LOAD will load the program from a disk.
7. Each new unformatted disk must be **FORMAT**ted. The **DOS** system must be transferred to every new disk.
8. The program name must be less than 9 characters, start with a letter, and contain no special symbols.
9. Saving a program on a disk is much faster than saving it on a tape. Also subsequent loading from disk is faster.
10. a) Cassette

1. Place a tape into the recorder & rewind it.
2. Set the tape counter to 0, then advance the counter 2 digits beyond the end of the last program.
3. Type **CSAVE** & press **RETURN**.
4. Press **RECORD** & **PLAY**, then press **RETURN**.

b) Disk

1. Be sure to turn on the disk drive and T.V.
2. Insert a System Disk into the disk drive.
3. Turn on the Computer.
4. Type: **SAVE "D1: PROGRAM NAME"**, & press **RETURN**.

CHAPTER 7

CHECKPOINT:

1. The corrected statements are:

```
10 INPUT A,B
15 INPUT A      or PR."WHAT IS YOUR AGE":I.A
20 INPUT A,A$
25 INPUT X,Y,Z$
30 INPUT L,M,N
```

2. ERIC 1,3,6
10 12

3. 10 PR."WHAT IS YOUR NAME":I.N\$

THINK:

(a) READ program

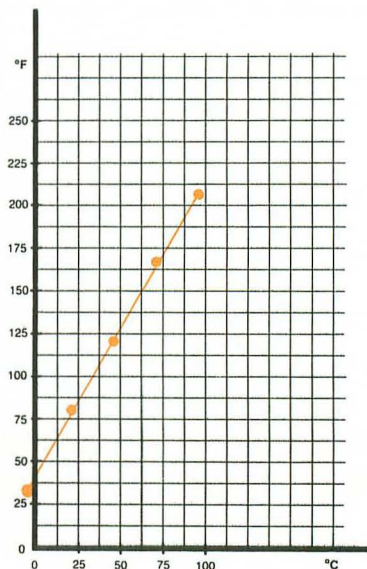
```
20 DATA 4,5,6
RUN
SUM IS 15
```

(b) INPUT program

```
RUN
?4,5,6
SUM IS 15
```

EXPERIMENT:

°C	0	25	50	75	100
°F	32	77	122	167	212

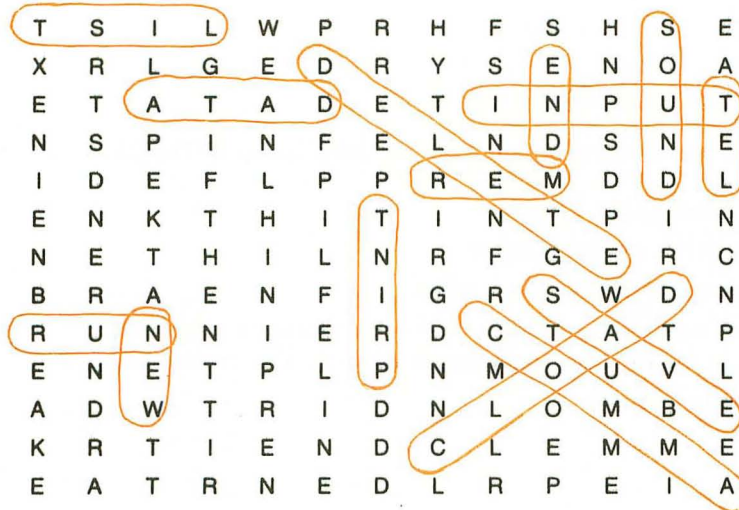


DID YOU KNOW:

Water boils at 212° F and freezes at 32° F.

FUN TIME

- | | | | |
|-----------|-----------|-----------|---------|
| 1. LIST | 2. CLOAD | 3. DELETE | 4. RUN |
| 5. LET | 6. SOUND | 7. SAVE | 8. END |
| 9. DATA | 10. REM | 11. COMMA | 12. NEW |
| 13. PRINT | 14. INPUT | | |



EXERCISES:

1. PRINT“\” can be included in a program. It then clears the screen only when the program is run. Whenever the **SHIFT** & **CLEAR** keys are pressed the screen is cleared immediately.
2. The program computes your age in year 2001.
3. NEW


```
5 DIM N$(6),P$(17)
10 PRINT"KNOCK KNOCK, WHO IS THERE" :INPUT N$
20 PRINT N$;" WHO"
30 INPUT P$
40 PRINT "YOU MUST BE PULLING MY LEG!"
```
4. Press the **SHIFT** & **CLEAR** keys.
Type **LC**, and press **RETURN**.
5. NEW


```
10 PR. "THREE NUMBERS" :I. A,B,C
20 PR. "THEIR PRODUCT IS ";A*B*C
30 END
```

The END statement is optional.

CHAPTER 8

COMPUTER QUIZ:

NOPE, THE CORRECT ANSWER IS 24

EXPERIMENT:

1. Replace Lines 20, 30, and 40.
2. The following lines replace Lines 10, 30, and 40 of the computer quiz program:


```
10 PRINT "YOUR THREE NUMBERS ": INPUT A,B,C
30 IF S=A+B+C THEN 60
40 PRINT "NOPE, THE SUM IS ";A+B+C
```

CHECKPOINT:

1. There are six comparison symbols.
2. Branch to another part of the program if the condition in the **IF-THEN** is true.
3.
 - a. Line number missing after **THEN**.
 - b. The IF statement has no line number.
 - c. $< > =$ is an incorrect comparison symbol.
 - d. This statement has an error in logic. If $W = 10$ transfer is made to Line 45. If W is not equal to 10, the run also continues on Line 45. That makes no sense!
 - e. Must use the equal sign $=$.

STRINGS AGAIN:

String variable **A\$**; the character string is
 "I WAS BORN IN 1776"

MORE PRACTICE WITH IF-THEN:

YOU LOST YOUR COOL

CHALLENGE:

If you type **TRUE** the program is still OK.
 If you type in **FALSE** the Computer responds:

YOU ARE RIGHT ON
 YOU LOST YOUR COOL

The **END** between Lines 50 and 60 is needed. Without Line 55 the Computer's response makes no sense.

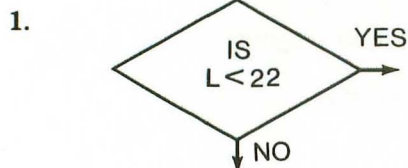
FUN TIME:

<u>C</u>	<u>O</u>	<u>M</u>	<u>P</u>	<u>U</u>	<u>T</u>	<u>E</u>	<u>R</u>	<u>S</u>	<u>A</u>	<u>R</u>	<u>E</u>	<u>F</u>	<u>O</u>	<u>R</u>	<u>K</u>	<u>I</u>	<u>D</u>	<u>S</u>
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19

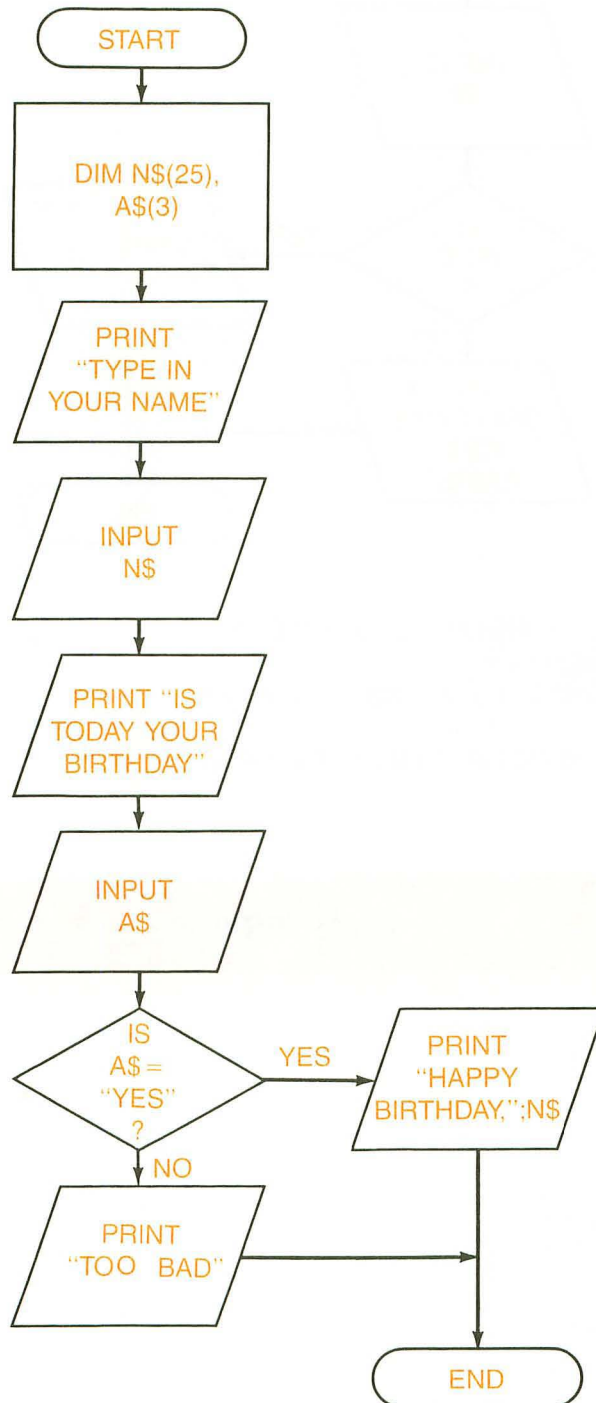
- a. **CLEAR** b. question mark c. comparison symbol
- d. unequal e. **LET** f. **RETURN** g. string h. slash
- i. **READ** j. **END** k. **IF THEN** l. condition m. **RUN**
- n. keyboard o. **LIST** p. **DELETE** q. less

*COMPUTERS ARE FOR KIDS is a registered trademark of:
 National Computer Camps Inc.
 Box 585
 Orange, Connecticut 06477

EXERCISES:



2. IF L < 22 THEN 40
3. 60 IF K <> 63 THEN 65
64
4. 20 IF A < 18 THEN 50
40 END
- 5.

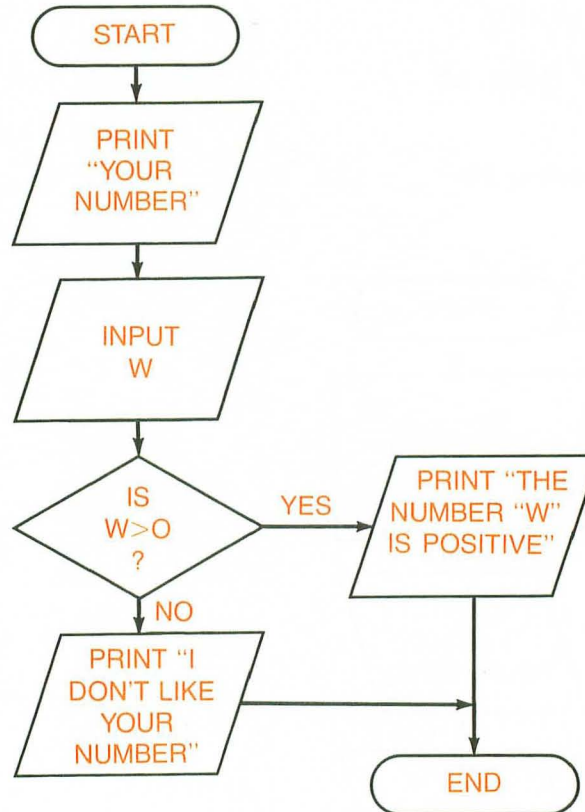


6. Quotation marks around LINCOLN are missing and Line 50 is missing:

```

20 IF N$="LINCOLN" THEN 50
50 PRINT "HI ABE"
  
```

7.



8. NEW

```

10 PRINT "YOUR NUMBER ": INPUT W
20 IF W > 0 THEN 40
30 PRINT "I DON'T LIKE YOUR NUMBER"
35 END
40 PRINT "THE NUMBER ";W;" IS POSITIVE"
50 END
  
```

CHAPTER 9

THINK:

1. 1
2. 2
3. Press the **BREAK** key.

YOUR TURN:

```

Type in 2 numbers
77,9
7+9 = 16
Type in 2 numbers
?
(etc..)
  
```

EXPERIMENT: Without Line 45 a no answer to the IF will display JUST RIGHT and TOO BIG.

CHALLENGE:

```

NEW
10 PAY = 1 : SQUARE = 1
20 PRINT SQUARE; " "; PAY,
30 SQUARE = SQUARE + 1
40 PAY = PAY * 2
50 IF SQUARE <= 64 THEN 20
60 REM A CHESSBOARD HAS 64 SQUARES
70 END
    
```

EXERCISES:

1. a. 1

b. 0

c. What is your name

?Atari

Hello Atari

?

"

"

"

(Notice GOOD-BYE is never printed.)

d. 0

2

4

6

8

10

e. 4

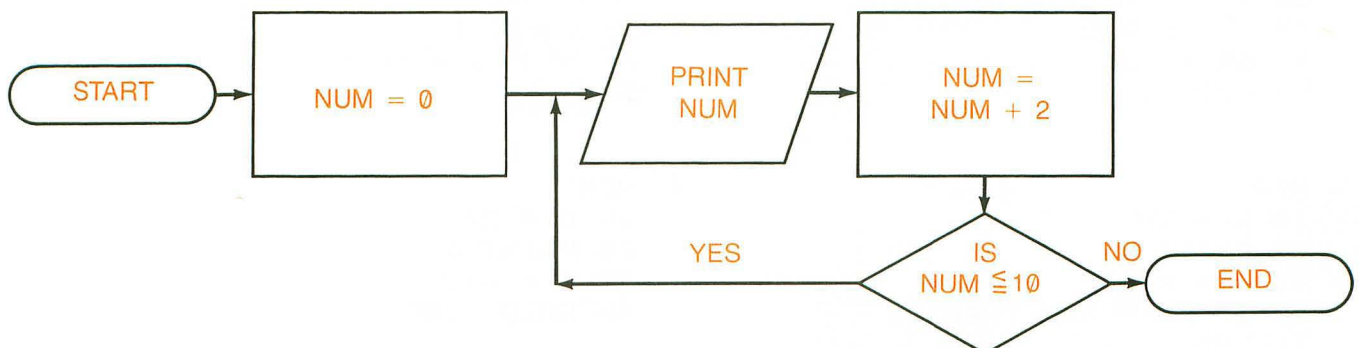
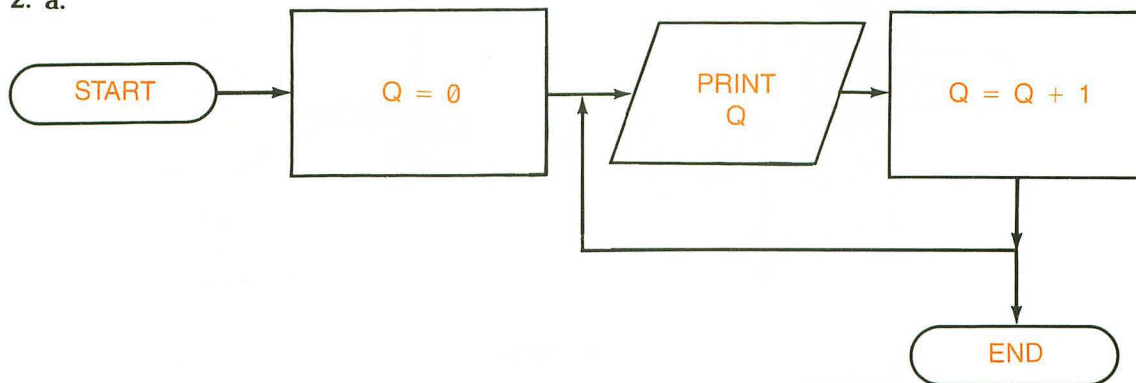
f. 3+3 = 6

8+8 = 16

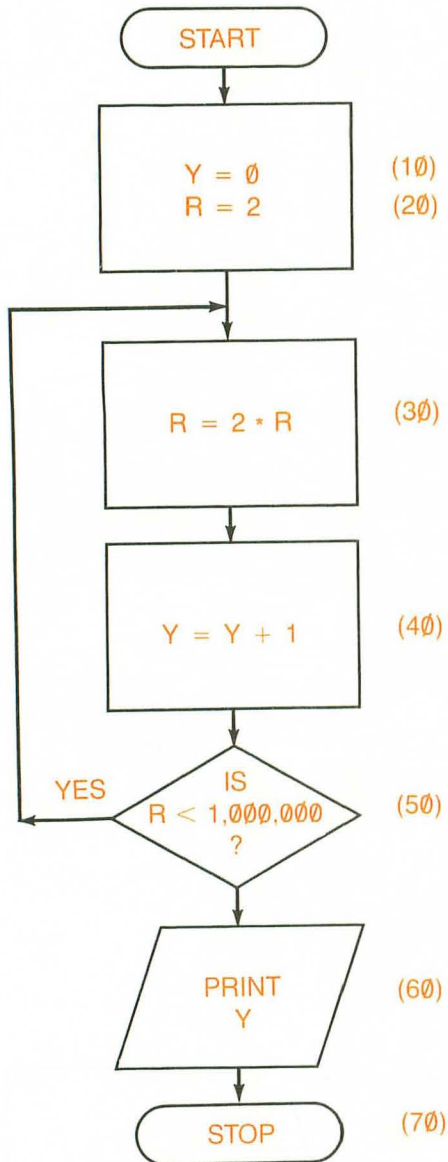
12+12 = 24

NO MORE DATA

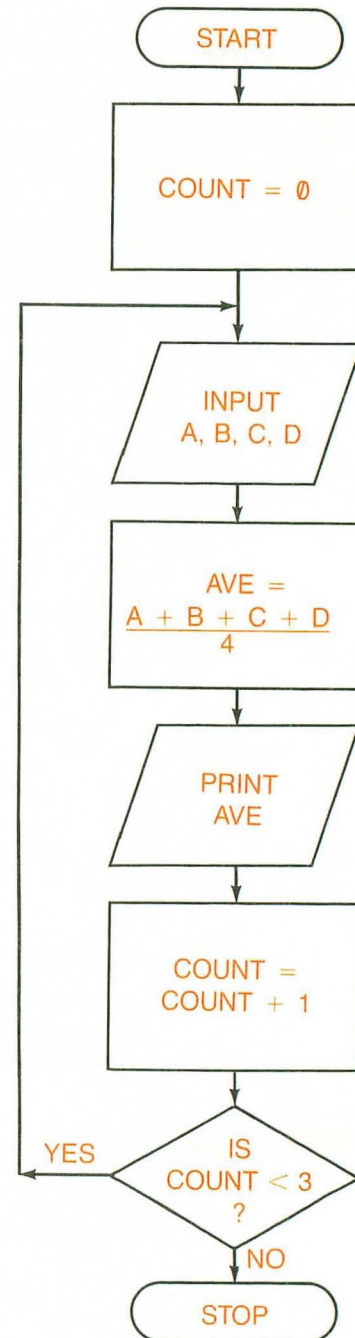
2. a.



2. b.



2. c.



3. a. NEW

```

5 PR. "ENTER 5 NUMBERS"
10 INPUT A,B,C,D,E
20 AVE = (A+B+C+D+E)/5
30 PRINT AVE
40 GO TO 5
50 END
  
```

b. NEW

```

10 K = 100
20 PRINT K
30 K = K-1
40 IF K > 0 THEN 20
50 END
  
```

c. NEW

```

10 K = 20
20 PRINT K
30 K = K+5
40 IF K <= 500 THEN 20
50 END
  
```

d. NEW

```

10 Q = 20
20 PRINT Q
30 Q = Q+2
40 GOTO 20
  
```

e. NEW

```
10 NUM = 11
20 PRINT NUM
30 NUM = NUM+2
40 IF NUM <= 27 THEN 20
50 END
```

f(i). NEW

```
10 Y = 0
20 R = 2
30 R = 2*R
40 Y = Y+1
50 IF R < 1000000 THEN 30
60 PRINT Y
70 END
```

f(ii). If the rabbit population increases by 50% each year change Line 30:

```
30 R = R+.5*R
```

CHAPTER 10

THINK:

```
30 PRINT L;" ";9*L
20 FOR L=4 TO 7
```

YOUR TURN:

```
L      5,6,7
O      5,7,9
V      1,4,7,10
E      10,20,30,40,50
M      10,9,8,7,6,5
Y      10
10 FOR A=101 TO 104
10 FOR T=1 TO 3.5 STEP 0.5
10 FOR A=15 TO -10 STEP -5
10 FOR R=10 TO 0 STEP -2
10 FOR I=.6 TO 1.0 STEP .1
```

CHALLENGE:

1. 10 times
2. 1 time
3. 1 time
4. 4 lines
5. No — because of the trailing comma in Line 20.
6. Yes — because there is no trailing comma in Line 40.

FUN TIME:

Counter variable outer loop is L.

Counter variable inner loop is **PAUSE**.

Loop of Lines 40-41 is not nested.

Loop of Lines 60-61 is nested in the loop of Lines 50-65.

THE TWELVE DAYS OF CHRISTMAS:

78 gifts.

EXERCISES:

1.a. RUN

b. RUN
4 3 2 1 0

c. RUN
5 25
6 36
7 49
8 64

2. RUN
ROW ROW ROW YOUR BOAT

3.a. NEW
10 REM STARS AND DASHES
20 FOR A=1 TO 3
30 PRINT "***_""
40 PRINT "____**"
50 NEXT A

b. NEW
10 REM TABLE OF SQUARES & CUBES
20 FOR B=1 TO 4
30 PRINT B;" " ";B*B;" " ";B*B*B
40 NEXT B

4.

K	L	K*L
1	4	4
1	5	5
2	4	
2		

5.a. IF-THEN

```
NEW
10 K=1
20 PRINT K;" ";
30 K = K+2
40 IF K<20 THEN 20
50 END
```

b. IF-THEN

```
NEW
10 K=3
20 PRINT 1/K;" ";
30 K = K+1
40 IF K <= 10 THEN 20
```

FOR-NEXT

```
NEW
10 FOR K=1 TO 20 STEP 2
20 PRINT K;" ";
30 NEXT K
40 END
```

FOR-NEXT

```
NEW
10 FOR K=3 TO 10
20 PRINT 1/K;" ";
30 NEXT K
```

6. Add the following lines to the Christmas song:

15 T = 0


```

65 T = T+G
85 PR."TOTAL NUMBER OF GIFTS FOR 12 DAYS IS ";T

```

7. NEW

```

10 TOTAL=1:PAY=1
20 FOR SQUARE=2 TO 64
30 PAY=PAY * 2
40 TOTAL=TOTAL + PAY
50 NEXT SQUARE
60 PRINT TOTAL

```

CHAPTER 11

CHECKPOINT:

SMALLEST NUMBER	LARGEST NUMBER
0	9.999999999
5	14.999999999
10	19.999999999
4	23.999999999
5	6.999999999

THINK:

5
7
0
9
3
5

YOUR TURN:

SMALLEST NUMBER	LARGEST NUMBER
0	9
0	14
0	19
1	10
1	20
6	20
3	20
1	7

CHALLENGE

```

20 PR. INT(6*RND(0))+1
26 Y=INT(6*RND(0))+1
30 PR. X;" ";Y;" ";X+Y

```

EXPERIMENT:

Blanks now correspond to random numbers 1, 3, 4, and 5;

\$'s correspond only to the random number 2.

DISCOVERY:

80 PR. "THE AVERAGE OF ALL 300 NUMBERS IS ";S/300

The more often we run the program, the closer the average gets to the expected value 2.

EXERCISES:

1. a. PR. INT(2*RND(0))+1
 b. PR. INT(6*RND(0))+1
 c. PR. INT(6*RND(0))+1+INT(6*RND(0))+1
2. a. PR. INT(6*RND(0))
 b. PR. INT(6*RND(0))+1
 c. PR. INT(100*RND(0))+1
 d. PR. INT(16*RND(0))+5
 e. PR. INT(2*RND(0))+7
 f. PR. INT(3*RND(0))
3. NEW
 10 FOR M=1 TO 8
 20 IF INT(2*RND(0))+1=2 THEN 35
 25 PR. "HEADS",
 30 GO TO 40
 35 PR. " TAILS",
 40 NEXT M
4. NEW
 10 FOR M=1 TO 4
 20 X=INT(3*RND(0))+1
 25 IF X=1 THEN 55
 30 IF X=2 THEN 45
 35 S=S+50
 40 GO TO 60
 45 S=S+20
 50 GO TO 60
 55 S=S+5
 60 NEXT M
 65 PR. "YOUR SCORE ";S
5. NEW
 10 FOR K=1 TO 5
 15 PRINT "TYPE IN YOUR GUESS":INPUT G
 20 A=INT(6*RND(0))+1
 25 B=INT(6*RND(0))+1
 30 IF G=A+B THEN 45
 35 PRINT "SORRY THE ROLL WAS ";A+B
 40 GOTO 55
 45 PRINT "CONGRATULATIONS!"
 50 S=S+50

```
55 NEXT K
60 PRINT "YOUR FINAL SCORE IS ";S
```

CHAPTER 12

CHECKPOINT:

POS.4,6	four	six
POS.8,15	eight	fifteen
POS.20,10	twenty	ten
POS.9,4	nine	four
POS.15,0	fifteen	zero
POS.38,1	thirty-eight	one
POS.30,22	thirty	twenty-two

YOUR TURN:

1. NEW
10 POS.10,15 : PR."*"
20 POS.16,15 : PR."*"
30 POS.10,21 : PR."*"
40 POS.16,21 : PR."*"
2. NEW
10 FOR X=0 TO 39
20 POS.X,4 : PR."#"
30 NEXT X
40 END
3. NEW
10 FOR R=0 TO 23
20 POS.19,R :PR."\$"
30 NEXT R
40 END

EXPERIMENT:

The results will vary depending on your input, but a sample follows:

```
What is your first name?CARA
HAVE A
GOOD DAY
CARA
```

*HAVE is orange,
A is dark blue.
GOOD is light green,
DAY is red.
CARA is orange.*

EXPERIMENT:

1. NEW
10 GR. 1+16
15 FOR R=10 TO 12
20 POS.9,R : PR.#6;"*"
25 NEXT R
30 GO TO 30
2. NEW
10 GR.17
20 POS.8,11
30 PR.#6;"***"
40 GOTO 40
50 END

CHALLENGE:

Change Line 30 to inverse characters.

30 POS. DOT, DOT:FR.#6;"#"

The # is inverse.

EXERCISES:

1. a) Clears the screen.
- b) Prints HI (white) in columns 10 & 11 of row 10.
- c) Prints blanks in columns 10 & 11 of row 10 (erases b).
- d) Prints HI (white) in the upper left corner of the screen.
- e) Prints HI (white) in the text window.
- f) Prints HI (white) in the text window.
- g) Prints a double width HI (orange) in the graphics window.
- h) Prints a double wide — double tall HI (light green) in the graphics window.

2.	Corners of entire screen		Corners of graphics window		Corners of text window
a.	0,0 39,0 0,23 39,23		****		0,0 39,0 0,23 39,23
b.	0,0 19,0 0,23 19,23		0,0 19,0 0,19 19,19		0,20 19,20 0,23 19,23
c.	0,0 19,0 0,11 19,11		0,0 19,0 0,9 19,9		0,10 19,10 0,11 19,11
d.	0,0 19,0 0,23 19,23		0,0 19,0 0,23 19,23		****
e.	0,0 19,0 0,11 19,11		0,0 19,0 0,11 19,11		****

3. Change the type of character — upper case to lower case or to inverse video.
4. Add 16 to the 1 in GR.1 — this gives GR. 17
- 5.a. Clears screen — pause
DANGER AHEAD — at column 10 row 8 — pause
this continue to flash on and off until you press **BREAK**

b. HELLO — in text window

HELLO — at column 7 row 4 — double wide
clear screen

HELLO — at column 7 row 4 — double tall double wide

Notice the same **POS.** numbers result in different locations when you change modes.

6.a. change Line 20:

```
20 F.A=1 TO 400:N.A
```

b. change Lines 30 and 40:

```
30 POS.4,7: PR.#6;"GOODBYE"
40 F.A=1 TO 500:N.A
omit Line 50
```

CHAPTER 13

YOUR TURN:

No, the pixels are at different positions. In **GR.4** the pixels are at the corners of the screen. In **GR.6** the pixels are framing the upper left quarter of the screen.

CHECKPOINT:

1. GR.4+16 GR.6+16
2. 0,0
3. Y=0, and X varies from 0 to 79.
4. Y=0, and X varies from 0 to 159.
5. X=79, and Y varies from 0 to 39.
6. X=159, and Y varies from 0 to 79.
7. 80 print positions along X and 40 print positions along Y.
8. 160 print positions along X and 80 print positions along Y.

CHALLENGE:

a) NEW
10 GR.6:C.1
20 PL.0,20:DR.159,20
30 END

b) NEW
10 GR.6:C.1
20 PL.20,0:DR.20,79
30 END

c) NEW
10 GR.6 : C.1
20 PL.20,10 : DR.139,10
30 DR.139,60 : DR.20,60
40 DR.20,10

d) NEW
10 GR.6 : C.1
10 PL.20,10 : DR. 20,29
30 DR.59,29 : DR.20,10
40 END

e) NEW

10 GR. 6 : C.1

20 PL.20,40 : DR.140,40 : DR.140,60 : DR.20,60

30 DR.20,40 : DR.80,10 : DR.140,40

THINK:

1. L = length of the line.

B = row containing the line.

2. Line 30. The longest line is 159 graphics blocks long.

3. Start at block position A and end at block position A+L.

4. The **TRAP** avoids the cursor out of range error. It occurs when the cursor goes off the screen; when $A+L > 159$.

EXPERIMENT

The Computer lights pixels on the screen. It does not print the number nor the words.

FUN TIME

1 P	O	2 S	I	T	I	O	3 N				4 R	U	5 N		
R		E					E		6 T				E		
7 I	N	T		8 D	R	A	W	9 T	O				X		10 G
N		C						R		11 L	I	S	T		R
T		O		12 D		13 O	S	A	V	E					A
	14 C	L	O	A	D			P		T		15 S	T	E	P
		O		T		16 I	17 F		18 D			O			H
		19 R	20 E	A	21 D		O		22 I	N	P	U	T		I
			N		O		23 R	E	M			N			C
24 L	O	A	D		S					25 R	N	D			S

EXERCISES:

1. GR. 4

2. GR. 7

3. GR. 7+16 or GR. 23. There is no text window for READY.

4. NEW

10 GR. 4 : C.1

20 PL.10,10 : DR.69,10

30 DR.69,29 : DR.10,29

40 DR.10,10

5. NEW

10 GR.4 : C.1

20 F. R=10 TO 29

30 PL.10,R : DR.69,R

40 N. R

6. NEW

```
10 GR.7 : C.1
20 PL.80,10 : DR.70,20 : DR.55,25
30 DR.70,30 : DR.60,45 : DR.80,35
40 DR.100,45 : DR.90,30 : DR.105,25
50 DR.90,20 : DR.80,10
60 PR. " TWINKLE TWINKLE"
```

7. a) Draws a large HI.

b) Draws a flag in three colors.

c) Draws a line with more than 100 pixels, then erases all the pixels after position 100.

d) Draws a rectangle inside another rectangle, in different colors.

8. a. NEW

```
10 GR.7+16 : C.1
15 REM G
20 PL.25,0 : DR.5,0
30 DR.5,95 : DR.2,95
40 DR.25,80 : DR.15,80
45 REM O
50 PL.30,0 : DR.30,95
60 DR.55,95 : DR.55,0
70 DR. 30,0
75 REM O
80 PL. 60,0 : DR.60,95
90 DR.85,95 : DR.85,0
100 DR.60,0
105 REM D
110 PL.90,0 : DR.90,95
120 DR.100,95
130 DR.115,49 : DR.115,45
140 DR.100,0 : DR.90,0
145 REM Y
150 PL.120,0 : DR.135,45
160 DR. 150,0
170 PL.135,46 : DR.135,95
180 G. 180
```

b. NEW

```
10 GR.5+16
20 FOR T=1 TO 3
30 READ A,X,Y
40 C.A : PL.X,Y
50 FOR L=1 TO 3
60 READ X,Y
70 DR.X,Y
80 NEXT L
90 NEXT T
100 DATA 1,30,0,0,24
110 DATA 78,45,30,0
120 DATA 2,30,3,6,23
130 DATA 68,40,30,3
140 DATA 3,30,10,18,21
150 DATA 52,32,30,10
160 G. 160
```

APPENDIX

ATARI RESERVED WORDS AND ABBREVIATIONS

The following is an alphabetic directory of the reserved words, and their abbreviations. The period is required. The listing also includes the chapter in which each of the reserved words is introduced. This list includes only reserved words introduced in this book.

WORD	ABBREVIATION	CHAPTER	WORD	ABBREVIATION	CHAPTER
CLOAD	CLOA.	6	NEW		4
COLOR	C.	13	NEXT	N.	10
CSAVE		6	PLOT	PL.	13
DATA	D.	5	POKE	POK.	13
DIM	DI.	5	POSITION	POS.	12
DOS	DO.	6	PRINT	PR. OR ?	2
DRAWTO	DR.	13	READ	REA.	5
END		4	REM	R.	7
FOR	F.	10	RND		11
GOTO	G.	9	RUN	RU.	4
GRAPHICS	GR.	12	SAVE	S.	6
IF		8	SETCOLOR	SE.	3
INPUT	I.	7	SOUND	SO.	3
INT		11	STEP		10
LET	LE.	4	THEN		8
LIST	L.	5	TO		10
LOAD	LO.	6	TRAP	T.	9

ERROR MESSAGES

ERROR CODE NUMBER	ERROR CODE MEANING
2	Memory insufficient.
3	Value error.
4	Too many variables (128 maximum).
5	String length error.
6	Out of data error.
7	Number > 32767.
8	Input statement error.
9	Array or string DIM error.
10	Argument stack overflow.
11	Floating point overflow/underflow error.
12	Line not found.
13	No matching FOR statement.
14	Line too long error.
15	GOSUB or FOR line deleted.
16	RETURN error.
17	Garbage error.
18	Invalid string character.
19	LOAD program too long.
20	Device number error.
21	LOAD file error.
128	BREAK abort.
130	Nonexistent device.
137	Attempt to ENTER a SAVEd file.
141	Cursor out of range.
147	Insufficient RAM .
170	File not found.

GRAPHICS MODES

GRAPHICS MODE	SCREEN SIZE		# OF COLORS	SE. REGISTERS
	WITHOUT TEXT WINDOW	WITH TEXT WINDOW		
0 and all text windows.	***	40 × 24	3	1 character luminance 2 background 4 border
1	20 × 24	20 × 20	5	0 upper case letters 1 lower case letters 2 upper case inverse 3 lower case inverse 4 background/border
2	20 × 12	20 × 10	5	same as GR.1
3	40 × 24	40 × 20	4	0 lines 1 lines 2 lines 4 background/border
4	80 × 48	80 × 40	2	0 lines 4 background/border
5	80 × 48	80 × 40	4	same as GR.3
6	160 × 96	160 × 80	2	same as GR.4
7	160 × 96	160 × 80	4	same as GR.3
8	320 × 192	320 × 160	3	same as GR.0
9	80 × 192	***	1	4 1 color/16 luminances
10	80 × 192	***	9	use POKE 704 through 712
11	80 × 192	***	16	4 16 colors/1 luminance

COLOR CHART

NUMBER	COLOR
0	BLACK
1	GOLD
2	ORANGE
3	RED-ORANGE
4	PINK
5	VIOLET
6	PURPLE-BLUE
7	BLUE
8	BLUE
9	LIGHT BLUE
10	TURQUOISE
11	GREEN-BLUE
12	GREEN
13	YELLOW-GREEN
14	ORANGE-GREEN
15	LIGHT ORANGE

SOUND CHART

	NOTES	PITCH VALUE
HIGH	C	29
NOTES	B	31
	A [#] or B ^b	33
	A	35
	G [#] or A ^b	37
	G	40
	F [#] or G ^b	42
	F	45
	E	47
	D [#] or E ^b	50
	D	53
	C [#] or D ^b	57
	C	60
	B	64
	A [#] or B ^b	68
	A	72
	G [#] or A ^b	76
	G	81
	F [#] or G ^b	85
	F	91
	E	96
	D [#] or E ^b	102
	D	108
MIDDLE C	C [#] or D ^b	114
	C	121
	B	128
	A [#] or B ^b	136
	A	144
	G [#] or A ^b	153
	G	162
	F [#] or G ^b	173
	F	182
LOW NOTES	E	193
	D [#] or E ^b	204
	D	217
	C [#] or D ^b	230
	C	243

SUBJECT	PAGE	SUBJECT	PAGE
A			
abbreviations	10, App. I	counters	80, 85
addition	8	crossword puzzle	11, 145
ADVANCE button	44	CSAVE	44
area program	23, 33	CTRL key	4, 13, 21
argument	109	arrow	13
Arithmetic operations	8	CAPS LOWR	15
asterisk	8	DELETE BACK S	13
Atari key	4	INSERT	15
averaging	36, 116	cursor	13, 122
B		D	
background	17	DATA	33, 39
BACK S key	3, 6	DELETE key	3
backup	48	degrees	
BASIC	8	Celsius	60
BASIC cartridge	3	Fahrenheit	60
BASIC instruction	8, App. I	digital counter	43
BLAST OFF	98	dice	112
border	17	DIM	35
BREAK key	79	directory	50
C		disk	47, 51
CAPS LOWR key	15	DOS	47
capitals	15	drive	47
cassette	1	format	48
ADVANCE button	44	load	50
loading a program	44	master	47
PLAY button	44	program name	50
RECORD button	44	save	49
saving a program	43	system	49
Celsius	60	display	11
Centimeters	26, 65	division	8
chaining	19, 56	slash	8
Christmas song	101	dollar sign	34
CLEAR key	6, 35	do-nothing loops	98
clear screen	57, 144	DOS	47
CLOAD	44	menu	48
coin toss	112	DRAWTO	138
colon	19, 56	E	
color	16, App. IV	editing	15
COLOR	138	END	18, 25, 29
column	123	equal to	68
comma		erase	13
DATA	33	line	13
DIM	35	screen	35, 57
READ	33	error messages	13, App. II
PRINT	27, 30, 36	ERROR 6	81
trailing	95	error trap	82
comparison symbols	68	ESC key	57
COMPUTER MAGIC	57	ETCH-A-SKETCH	159
COMPUTER QUIZ	69	F	
conditional transfer	83	Fahrenheit	60
control (see CTRL)		FIREWORKS	157
coordinates	121	flowchart	66
COUNTDOWN	98	FOR/NEXT	93, 102
counter variable	102	body	102

SUBJECT	PAGE	SUBJECT	PAGE
counter variable	102	LINES GALORE	141
STEP	96	line numbers	25, 29
format	48	LIST	35, 39
four color mode	142	living room program	23, 33
FORTUNE WHEEL	114	LOAD	50
FUN TIME	11, 20, 38, 45, 62, 73, 98, 145	loop	79, 93
G		body	102
games		do-nothing	98
see programs		infinite	79
GOTO	79	nested	98, 103
IF	84	lower case	15
READ	81	LPRINT	11
GRAPHICS	125, 144	M	
graphics blocks	141	margins	95
graphics characters	15	MASTERFIND	156
graphics modes	121, App. III	master disk	47
0	121	menu	47
1	125, 128	messages	
2	125, 130	error	13, App. II
3	142	print	9
4	135	minus	8
5	142	modes (see graphics modes)	
6	135	monitor	1
7	142	multiplication	8
8	147	asterisk	8
graphics window	126	star	8
greater than	68	music	18
greater than or equal to	68	N	
H		nested loops	98, 103
heads	112	NEW	25
HI-LOW program	45	NEXT	93
I		not equal to	68
IF-THEN	68	numerical variables	34
loops	84	initial values	30, 40
strings	71	O	
infinite loop	79, 138	origin	123
INPUT	55, 59	P	
INT	109	parentheses	36
INSERT key	15	pixel	136
instruction	8, App. I	PLAY button	44
inverse video	143	play Computer	100
K		PLOT	138
keyboard	1, 2, 6	plus	8
L		POKE	95
less than	68	POSITION	122, 123
less than or equal to	68	column	123
LET	23, 25, 26	row	123
line		with PR.	129
change	39	PRINT	8, 11, 29
delete	39	clear screen	57, 144
number	29	comma	27, 30, 36
		messages	9, 10, 33

SUBJECT	PAGE	SUBJECT	PAGE
PR.	10	SHIFT key	4
PR. #6;	126	CAPS LOWR	15
?	10	CLEAR	4,6,35,57
quotation marks	9	DELETE	4, 6
semicolon	27, 30, 36	INSERT	15
program	24, 30	SHOOT THE DUCK	157
COMPUTER MAGIC	57	slash	8
ETCH-A-SKETCH	159	SNOWDAY	65
FIREWORKS	158	SOUND	18, App. V
HI-LOW	45	END	18
WITH A LYING COMPUTER	154	quality	18
MASTERFIND	155	SO.	21
names	50	tone number	18
SHOOT THE DUCK	157	voice number	18
TIMED ARITHMETIC DRILL	154	volume	18
WHEEL OF FORTUNE	160	SPACE BAR	3
Q		special symbols	3, 15
question mark		special keys	4
INPUT	55	star	8
PRINT	10	STEP	96
quotation marks	9	string variables	34
strings	34	IF	71
quotes (see quotation marks)		initial value	40
R		READ	36
random numbers	107	subtraction	8
READ	33, 39, 59	sums	100
ERROR 6	81	surprise numbers	107
GOTO	81	symbols	
READY	3, 7	comparison	68
RECORD button	44	operation	8
recorder	1	special	3
digital counter	43	system disk	48
load	44	SYSTEM RESET	4, 16
save	43	T	
REM	60	tails	112
remarks	60	tape	43
RETURN key	7	temperature conversion	60
RND	107	TERMITE TIME	145
row	123	text window	126
RUN	25	eliminate	129
S		THEN	68
SANTA	55	TIME OUT FOR OLD NEWS.....	6, 21, 30, 40, 62, 87, 102, 118, 131, 148
SAVE	49	toss	
screen	6	coin	113
background	17	dice	112
border	17	trailing comma	95
clear	35, 57, 144	trailing semicolon	95
display	11	transfer statements	
zones	27	conditional	83
semicolon	27, 30	unconditional	83
trailing	95	TRAP	81, 82
SETCOLOR	16	TRUE/FALSE	72
SE.	21	TWELVE DAYS OF CHRISTMAS	101
		two color modes	135

SUBJECT

PAGE

SUBJECT

PAGE

U

unconditional transfer	83
upper case	15

V

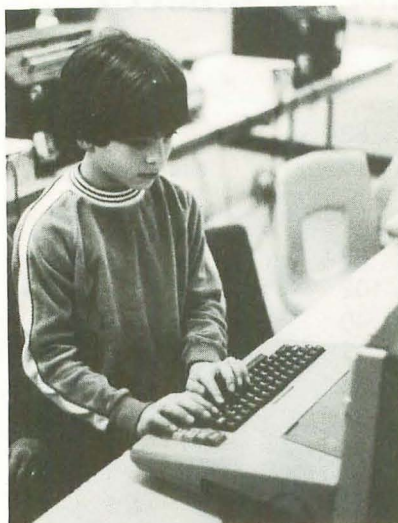
variables	23
counter	102
initial value	30, 40
names	24, 34
numerical	34
string	34

W

window	
eliminate text	129
graphics	126
text	126
word search	20, 38, 62

Z

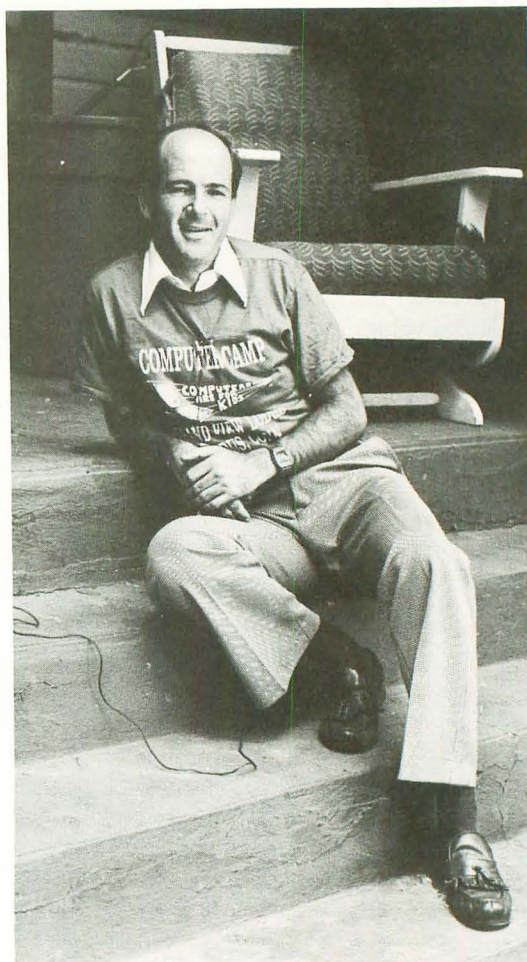
zero with slash	4
zone	27



Learning by doing



Small group instruction



DR. MICHAEL ZABINSKI — *Director
National Computer Camps®*

Michael P. Zabinski, Ph.D. is a professor at Fairfield University, Fairfield, Connecticut. He is founder and director of National Computer Camps®, the first computer camp in the world for youngsters. Dr. Zabinski is a consultant to public schools on computer usage in the classroom and author of programming books as well as educational materials for Radio Shack.

National Computer Camps, Inc.
Post Office Box 585
Orange, Connecticut 06477

ATARI® For Kids

From 8 to 80

Dr. Michael Zabinski is the founder and president of the National Computer Camps—the first computer camp in the world for youngsters. **E. Michael Scheck** has been teaching Computer Science at the high school level for more than six years. Mr. Scheck has also written programs for private businesses using microcomputers. This book is the result of their search for a suitable text for beginners to learn programming. It's a fun book written in a light and humorous style that anyone can use at school or at home. This easy to follow book covers the most important programming concepts and quickly allows the beginner the enjoyment of writing their own programs.

The reader is encouraged to try as many examples as possible, since programming is best learned by doing. Special features of this book are:

- CHECKPOINTS for review
- BRAIN FOOD for good learning
- TIME OUT FOR OLD NEWS for a reminder
- FUN TIME for recreation
- EXPERIMENTS to try new things
- CHALLENGES to stretch your mind
- GAMES for your collection
- EXERCISES for practice
- SOLUTIONS to exercises just in case

Howard W. Sams & Co., Inc.
4300 West 62nd Street, Indianapolis, Indiana 46268 U.S.A.

\$15.95/22294

ISBN: 0-672-22294-9