Commodore 64 Edition

COMPUTERS FOR KIDS

Sally Greenwood Larsen

A BASIC programming manual written just for kids.
Special section for teachers and parents.

COMPUTERS FOR KIDS

COMMODORE 64 EDITION

COMPUTERS FOR KIDS

COMMODORE 64 EDITION

SALLY GREENWOOD LARSEN

Creative Computing Press
Morris Plains, New Jersey

Computers For Kids—Commodore 64 edition

Adaptation by John J. Anderson. The section "MENU: The Easy Way to Use the C-64 Disk Drive" and the MENU program were written by John J. Anderson.

Library of Congress Cataloging in Publication Data

Larsen, Sally Greenwood.
Computers for kids.

Summary: An introduction to computers including machine operation, flowcharting, and BASIC programming. Also includes guidelines for parents or teachers using the book to teach children about computers.
1. Commodore 64 (Computer)—Programming—Juvenile literature. 2. Basic (Computer program language)
[1. Commodore 64 (Computer)—Programming. 2. Basic (Computer program language) 3. Computers. 4. Programming languages (Computers)] I. Title.
QA76.8.C64L37 1984 001.64'2 83-23165
ISBN 0-916688-63-1 (pbk.)

Creative Computing Press
39 East Hanover Avenue
Morris Plains, New Jersey 07950 USA

Manufactured in the United States of America.

86 85 84 987654321

For Chris, Erik, Jon, and Becky

TABLE OF CONTENTS

SECTION 1: What is a Computer?

When a caveman had work to do, he had no machines or tools to help him. He had to do it all by himself. Man has since invented many tools to help him with his work.

Instead of pounding with his hands, he now uses a hammer. The hammer lets him pound harder and longer than he could pound with his hands alone.

Man invented the telescope so that he could see farther into space. He can now see stars he did not know existed before he had the telescope to help his eyes.

Using his brain, man can remember information and solve problems.

Man wanted to invent a tool so that he could extend the use of his brain, so he invented the COMPUTER.

Just as a hammer can't do work without a person to hold it, a computer cannot do work without a person to run it and tell it what to do. This person is called a PROGRAMMER.

Even the best hammer cannot do all the different things our hands can do. And even the best computer cannot do everything our brains can do.

A computer cannot feel emotion. It cannot feel happy or sad, as we can.

A computer cannot combine ideas the way our brain can. It can't put two ideas together and take the best parts of each one to make a brand new idea.

But . . . a computer **can** do some of the simpler jobs our brains can do. And it can do some of them even faster than we can!

A computer can remember many more things than most of us can with just our brains, especially things like long lists of names or numbers. This information is kept inside the computer in the MEMORY. Computer programmers call this information DATA.

A computer can **compare** data to see if one thing is bigger than another, or smaller, or the same. It can also put things in order.

A computer can sort many pieces of data and put together the things that are alike.

And a computer can look in its memory to find the data a programmer wants, and print out that data on a video screen or a sheet of paper.

This book is about the Commodore 64 microcomputer. These are special directions for this computer. They will not work on all other kinds of computers.

The Commodore 64 is called a MICROCOMPUTER, because it is so small. Many businesses and universities have computers, too, but theirs have to do many more jobs than the Commodore 64, so they have to be much larger. Some of the biggest computers are so huge, that they fill an entire *room!*

The Commodore 64 uses a special computer language called BASIC. It is an easy language to learn, because it uses words we hear everyday.

Some bigger computers use languages called FORTRAN or COBOL. You might hear about other languages when you find out more about computers.

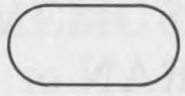
SECTION 2: Flowcharting

When you want the computer to do a job for you, you must break down the job into small steps, so the computer can understand what to do. One big job may have many small steps, and sometimes it is hard to keep track of all the steps.

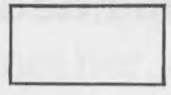
One way of keeping track is with a **flowchart.** A flowchart shows all the steps in a problem, shows what choices there are, and in what order the steps must be done.

On the next page is a flowchart showing all the little steps in a funny problem. The directions on this flowchart are things for you to do. They are not directions for the computer!

The shapes drawn around the steps show what **kind** of a step it is:



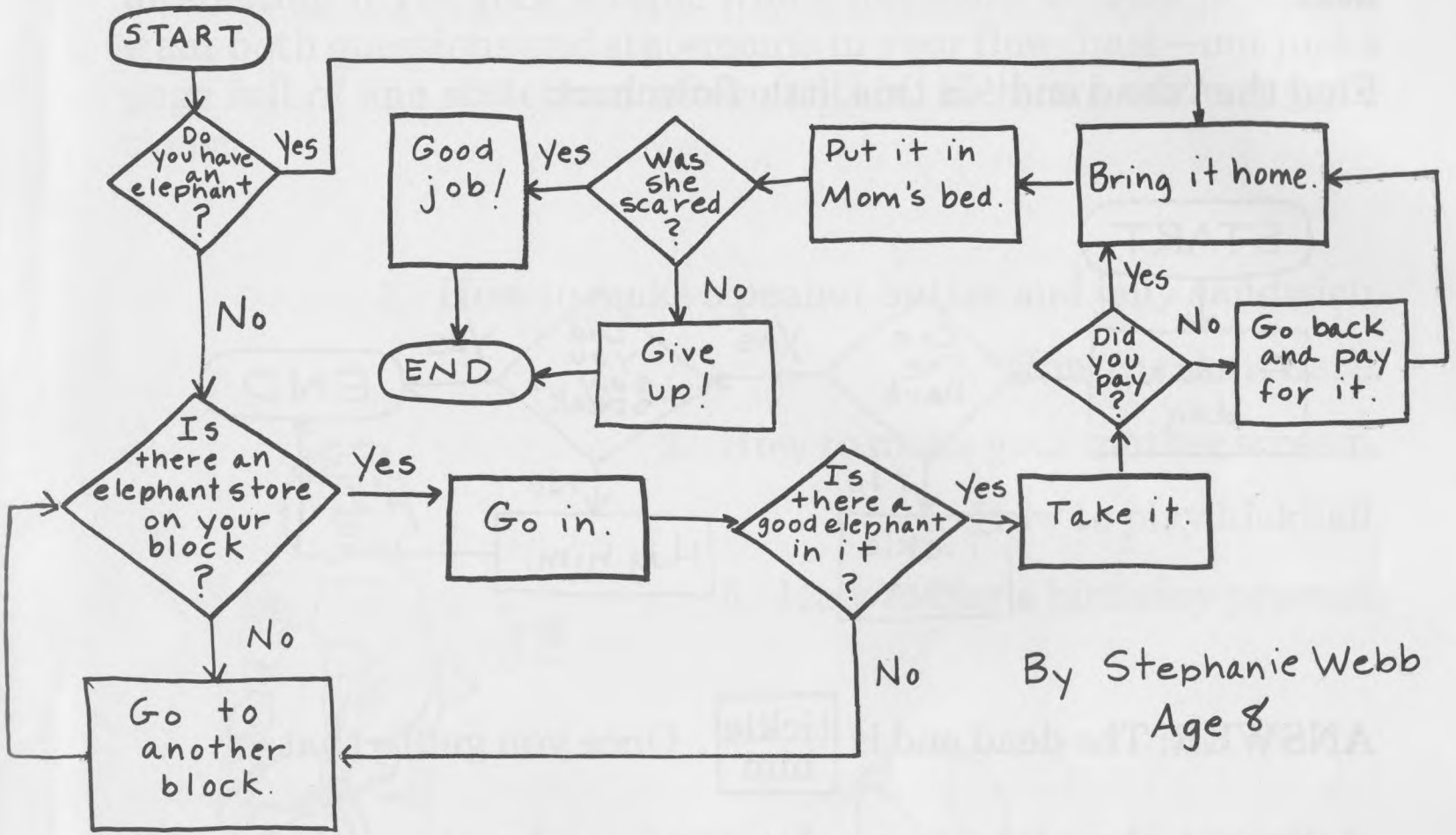
OVAL — for START or END.



RECTANGLE — for statements telling exactly what to do (you have no choice).

DIAMOND — for yes or no questions.

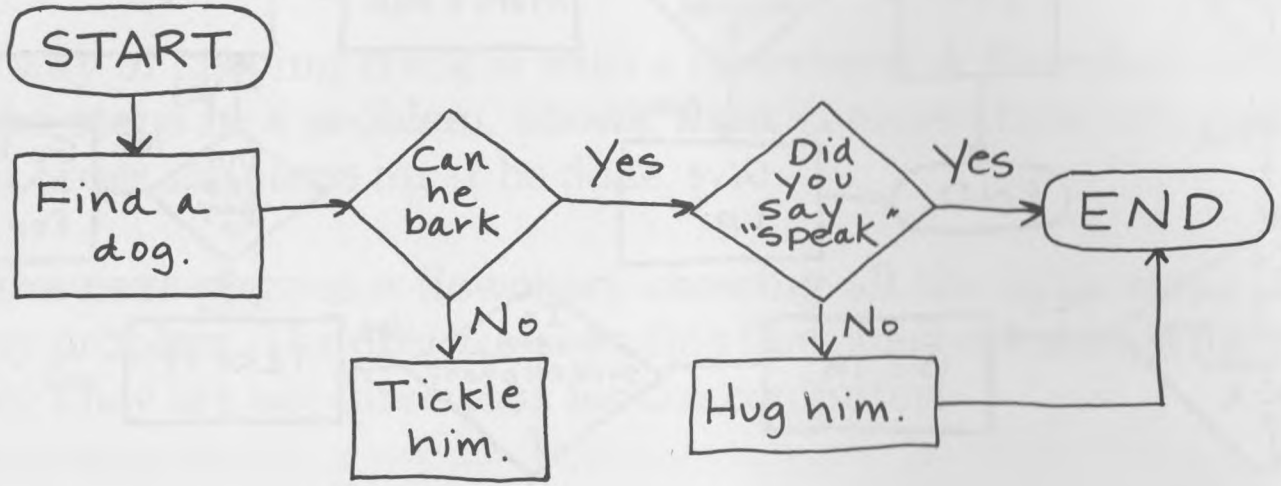
How to Scare Your Mom with an Elephant



The arrows on a flowchart show you what to do next. One arrow shows what to do if the answer is *yes*. The other arrow is for *no*.

There must be no "dead ends" in a flowchart. This means that there must always be an arrow showing what to do and where to go next.

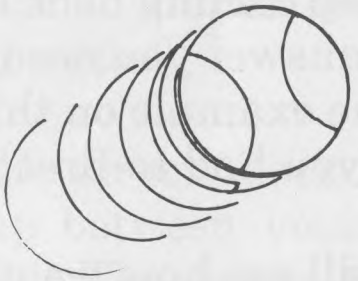
Find the "dead end" in this little flowchart:



ANSWER: The dead end is tickle him. Once you get to that statement, there is no arrow showing you where to go next.

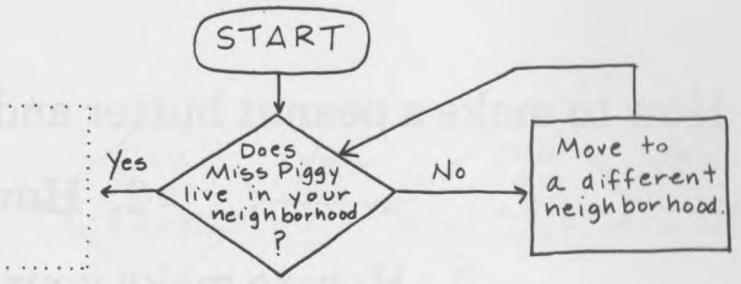
When you write your own practice flowcharts, pick a subject you know something about. Also, your flowchart will be much more interesting if you pick a topic which has some choices in it. You want both questions *and* statements in your flowchart—not just a page full of one statement after another. Here are some suggestions:

1. How to make a peanut butter and jelly sandwich.
2. How to take a bath.
3. How to make your mother scream.
4. How to play kickball.
5. How to buy a birthday present.



When the arrows in a flowchart make you do something over and over again, this is called a *DO-LOOP*. Here is an example: (I have only drawn *part* of this flowchart.)

How To Get A Date With Miss Piggy



If you follow the directions for this part of the flowchart, you will keep moving to a new neighborhood until you are living in Miss Piggy's neighborhood. This is called a **do-loop**.

In a do-loop, you keep coming back to the question you asked until you finally get the answer you need so you can go on to the rest of the flowchart. In the example on this page, in order to "get a date with Miss Piggy," you had to first move into her neighborhood.

In Section 8, you will see how we use flowcharts to help us write computer programs.

SECTION 3: Running the Computer Itself

The Commodore 64 computer has five basic parts:

The Keyboard: This looks like a regular typewriter keyboard. It has all the electronics for the computer inside it. You type in information and instructions for the computer on the keyboard.

The TV Screen: The information you type on the keyboard is printed out on the TV screen, so you can see what you are doing. The keyboard can be hooked up to almost any TV set. However, when you learn to make pictures on the screen, you will see that you need a color TV to make colored pictures. (If you use a black and white TV, the pictures you make will be black and white.)

The Power Supply: This is the larger black metal box which is hooked up between your keyboard and the wall outlet. The power supply box changes household electrical current so it is right for the computer.

The RF Modulator: This is the smaller black metal box. This box helps the C-64 send information to your TV set.

The Storage Device: This may be a cassette recorder or a disk drive. A cassette recorder is a special tape recorder which records programs you have written on your C-64. It uses regular tapes, like the ones you record music on. The cassette recorder is hooked to the keyboard with a special wire. The disk drive is connected to the keyboard with a different kind of wire, and if you want to, you can connect a cassette recorder and a disk drive to the computer at the same time. The disk drive stores programs on a small round piece of plastic with a special coating. This disk is protected by a square paper sleeve. A disk drive can do a lot more than a cassette recorder, but is a little more complicated to work with. You will find the directions for using the tape recorder or disk drive in Section 4.

The computer has a *Random Access Memory* (RAM for short). This means that the computer will hold data in its memory only as

long as the keyboard is left **on**, so it has electricity flowing through it. If you turn off or unplug the keyboard, you will lose your program. (Turning off only the TV won't matter.)

The first time you use the C-64, have an adult help you set up the machine and connect all the proper plugs. It is not hard to set up the computer, once you know how, but it is important that the job be done correctly, or you might ruin it.

IF YOU AREN'T SURE — ASK FOR HELP!!! It could take weeks to get your computer fixed if you break it.

Things To Remember

1. Before you start programming, you must plug in the Power Supply box, then turn on both the TV and the keyboard. (If this doesn't work, you probably forgot to plug them into the wall socket.) The TV volume should be turned all the way down, so you can't hear any sounds from the TV.

2. Take it easy with the keyboard—no pounding, please!

3. Keep your feet away from the electrical cords. If you accidentally kick out a plug, you may lose your program.

4. Good programmers never eat or drink while working. The computer will not work well if it is full of cookie crumbs, or has soda spilled between the keys!

5. You should never stick anything other than the proper connectors (plugs) or cartridges into the open slots in the back of the keyboard. If you reach inside and touch the wrong things, you could ruin your computer, or get an electrical shock. Otherwise, your computer is as safe as any other appliance you have at home.

6. Turn off both the TV and the keyboard when you are not using them. Remember to unplug the Power Supply box from the wall outlet, too. It should not be "on" when you're not using the computer.

7. Be extra careful when handling disks. Learn to hold them so that you never touch the little windows in the paper sleeves. Also be very careful not to bend or fold the disks. Disks can last a long time, but only if you treat them well. Otherwise you may lose programs you have saved on them.

Unless you need to save a program right away, skip Section 4 (about the cassette recorder and the disk drive), and go right to Section 5: Getting Ready to Program.

You can come back to Section 4 later.

SECTION 4: Saving Your Programs with a Cassette Recorder or a Disk Drive

You now have a computer program typed into the computer, and you want to save it on a cassette tape or disk drive. Follow these directions **carefully.**

Saving Your Program on a Cassette

1. Advance your cassette tape to the spot where you want to record your program. REMEMBER THE NUMBER OF THE TAPE RECORDER COUNTER! Just to be safe, make sure you have gone five numbers past the end of the last program on that tape.

2. Type SAVE "MY PROGRAM" on the computer. (In the quotes you can name your program whatever you like.) Press RETURN. The screen should say "PRESS PLAY & RECORD ON TAPE."

3. The C-64 is telling you to hold down the RECORD key on the cassette player and then hit PLAY. Both keys should stay down.

4. The C-64 should now tell you "OK" and "SAVING" while it is busy recording your program on tape.

5. When the C-64 is finished saving your program, it will say "OK" and "READY".

6. Now comes a very important step. Sometimes when you SAVE a program, the whole program might not end up on the tape. (Maybe the tape was bad, for instance.) The C-64 has a special command that tells the computer to double-check that your program is OK on the tape. First, you must rewind the tape back to the beginning of the program you just recorded.

7. Now type VERIFY "MY PROGRAM" and press RETURN. The C-64 will say "PRESS PLAY ON TAPE". It is telling you to . . .

8. Press the PLAY button on the tape recorder.

9. The C-64 will now say "OK" and "SEARCHING". It is searching the tape for the program you want to verify. It will tell you the name of any other program it finds, and when it finds the one you asked for, it will say "FOUND MY PROGRAM" (or whatever you named it) and "VERIFY."

10. The C-64 will say "OK" and "READY" if the program on the tape exactly matches the one you typed into the computer.

11. If you see "?VERIFY ERROR" that means the program is not OK on the tape, and you should save it again. Be sure to use a different spot on the tape this time, or try a different tape.

12. Your program is now recorded on the cassette. Make sure the location and name of your program are written down on the cassette label, so you can find it later. Saving the program on a tape does not erase it from the computer's memory. You must type NEW to do that.

If you have saved a program on a cassette, and now want to load it into your computer, follow these directions carefully:

Loading a Cassette Tape Program into the Computer

1. Advance your cassette to two or three numbers **before** the location of your program. (If your program is recorded at 85, for example, advance the tape to 82 or 83 on the counter.)

2. Type LOAD "MY PROGRAM" and press RETURN. Whatever you named your program goes in the quotes.)

3. The C-64 says "PRESS PLAY ON TAPE". You should push the PLAY button on the tape recorder.

4. When the C-64 is done loading the program into the computer's memory, it will say "READY".

5. If something went wrong, you will see the message "?LOAD ERROR." You will have to start over and try again to LOAD your program.

Your program should now be in the memory of the computer. You can LIST it or RUN it, just like any other program.

Using A Disk Drive

Working with a disk drive is a little tricky, but you can learn to use it. After you catch on, you won't want to use cassettes ever again. Disks are faster, and can hold lots more programs than cassettes. Also, the disk drive will locate your programs for you, so you won't have to keep track of counter numbers the way you do with cassettes.

You can only get into trouble with disks if you forget to do certain things that tell the disk drive what to do. If you make a mistake, you may not be able to load or save programs. You may even lose programs you've typed in. Until you've learned how to work the disk drive correctly, ask an adult who understands the disk drive to help you.

Getting Started With the Disk Drive

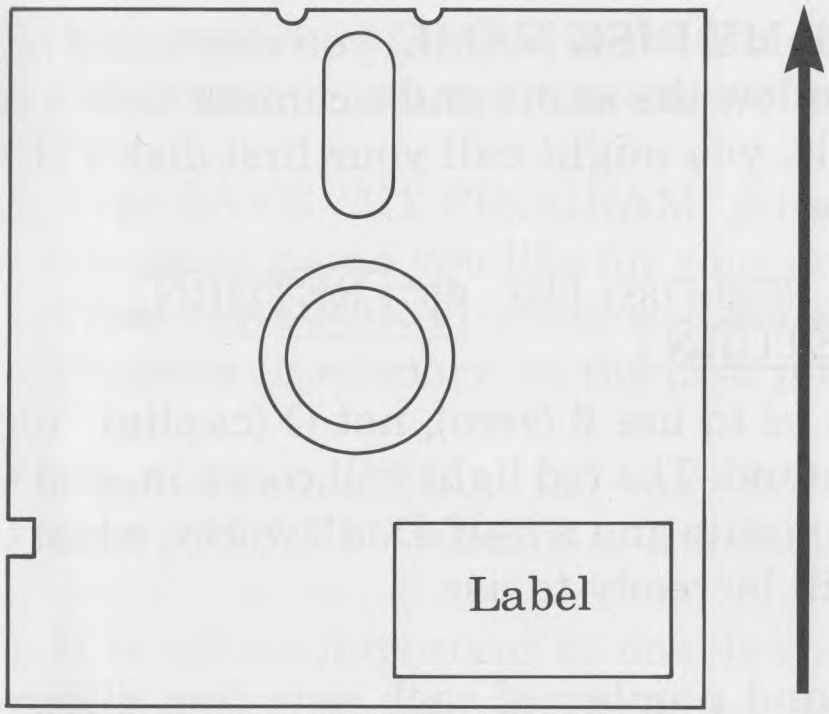
The first thing to remember is to turn the disk drive on before you turn on your C-64. If your C-64 is already on, turn it off, then turn it on again while the disk drive is turned on. This tells the computer that the disk drive is connected and ready to use.

Never turn the disk drive on or off while there is a disk inside it. This can destroy programs that you've saved on the disk. Make sure that the drive is empty before you click the power switch in the back on or off.

Once the drive is on, the green light on the front of it will come on. When the C-64 is turned on, the red light on the disk drive will come on for a few seconds, then go off. This means that the disk drive is ready.

The red light on the front of the disk drive means that the disk inside is spinning. Think of the red light as a traffic light: When it is red, don't put in or pull out a disk. Wait until the red light goes off before opening the drive door. Open the drive door by pressing on the black bar in the middle of the door. The door will snap open. It is a good idea to hold the bar with your thumb as you press it, so that it doesn't snap open too hard.

Now you are ready to put in a disk. Hold the paper sleeve on the side with the label on it, making sure that the label is facing up and that your fingers don't touch the shiny plastic disk through any of the windows on the sleeve. Slide the disk through the door of the drive, making sure that the big shiny window is the first part of the disk to go in.



Press this way into drive

Press the disk all the way into the drive. It will snap into place. Then push down on the black bar to shut the drive door.

Before you can use a blank disk to save your programs, it must be "formatted." This divides the disk up into "pages" that your C-64 can write on and read from. You must also give the disk its own name and a special identifying number. With a blank disk in the drive, type

```
OPEN 15,8,15, "NEWØ:MY DISK NAME, (TWO DIGIT NUMBER)" RETURN
CLOSE 15 RETURN
```

Instead of calling the disk MY DISK NAME, you can name your disk whatever you like. Follow the name and a comma with a two digit number. For example, you might call your first disk FIRST DISK, Ø1. You would type

```
OPEN 15,8,15, "NEWØ:FIRST DISK, Ø1" RETURN
CLOSE 15 RETURN
```

Don't forget the commas or to use Ø (zero), not O (capital "oh"), right after the NEW command. The red light will come on, and the disk will spin for about a minute and a half. Don't worry; when the disk stops spinning, it will be ready to use.

Always make the name and number of each new disk different from the ones you've used before. For example, you might call your second disk SECOND DISK, Ø2. If you give two disks the same name and number, things might get confusing.

WARNING! If you FORMAT a disk that already has programs on it, all those programs will be *erased!* Be careful!

Saving Your Program on a Disk

1. Make sure that the disk drive was turned on first, before the C-64.

2. Make sure that the disk in the drive has been formatted, and that the program you want to save is in the memory of the C-64.

3. Type SAVE "MY PROGRAM", 8 on the computer. (You can type whatever name you like for your program between the quotes.) Press RETURN . The number 8 tells your C-64 to save the program in memory on the disk instead of the cassette.

4. The C-64 should now tell you "SAVING MY PROGRAM" and then "READY" when it is finished.

5. It is not as important to double-check that your program has been saved on disk as it is when you save programs on tape. If you want to make sure that your program has been safely saved, type VERIFY "MY PROGRAM", 8 and press RETURN . Again, the 8 tells your C-64 to use the disk drive.

6. The C-64 will now say "SEARCHING FOR MY PROGRAM", and "VERIFYING". If the program has been saved correctly, the C-64 will print "OK" on the screen.

7. If you see "?VERIFY ERROR" it means that the program was not saved correctly, and you should try saving it again. Usually, the only time this will happen is when a disk is completely full. That happens only once in a long while. It is a good idea to keep a

blank disk that has already been formatted near the computer. That way, when you find out that a disk is completely full, you won't have to lose the program you want to save because you have to format a new disk.

Loading Your Program From A Disk

1. Make sure that the disk drive was turned on first, then the C-64.

2. Make sure that the disk in the drive has been formatted, and that it contains the program you want to load into the computer.

3. Type LOAD "MY PROGRAM", 8 on the computer. (Type whatever you've named the program between the quotes.) Press RETURN . The number 8 tells the C-64 to get your program from the disk drive.

4. The C-64 should now tell you "LOADING MY PROGRAM" and then "READY" when it is finished.

5. If something went wrong, you will see an error message. If it says "?FILE NOT FOUND ERROR" you are not typing in the right name for your file, or you've put the wrong disk into the drive.

Other Things to Learn about the Disk Drive

What if you can't remember the name of the program you've stored on the disk? To get a list of all the file names on a disk, type

LOAD "$",8 RETURN

Then type

LIST RETURN

This will list out the name and number of your disk, along with all the file names stored on it. If there are a lot of programs, they may go by too fast for you to read them. You can slow them down by pressing CTRL , or stop them by pressing RUN STOP .

WARNING! When you type

LOAD "$",8 RETURN

you are loading the disk file names into the computer's memory. This will erase any other program you have in the memory. Don't forget to SAVE the program in the memory before you LOAD the file names from the disk, or you will lose your program.

What if you want to get rid of a program? First, make sure you've decided that you really want to erase it. Next, type

```
OPEN 15,8,15, "SCRATCHØ:MY PROGRAM"  RETURN
CLOSE 15  RETURN
```

When you type this in, replace "MY PROGRAM" with the name of the file you want to erase. Remember to type Ø, not O, after the word SCRATCH.

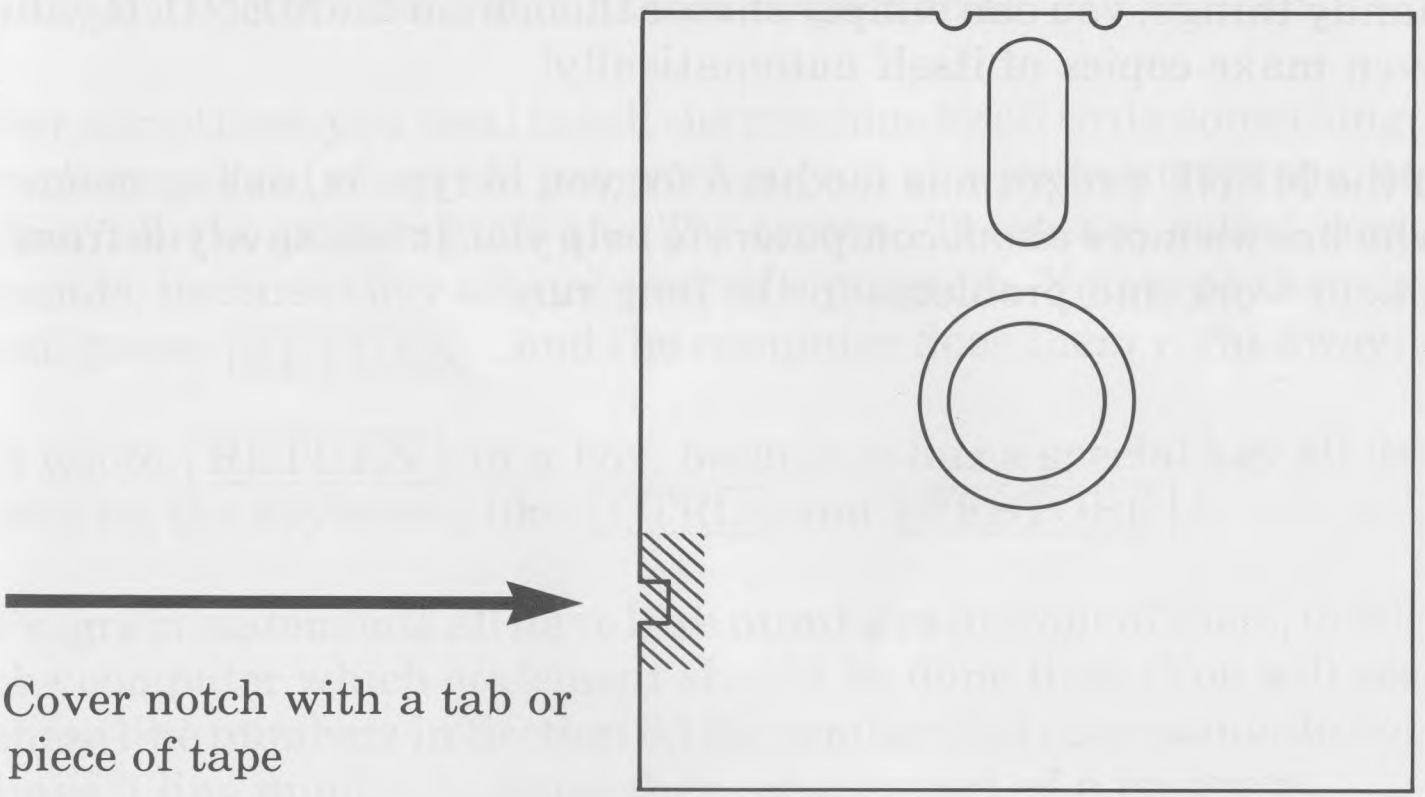
Sometimes you have made a program better, and want to save the new version of the program instead of the old one. To save a new version of a program (with the same file name as the old version) type

```
SAVE "@Ø:MY PROGRAM", 8  RETURN
```

The [@] key sits between the [P] and [*] on the keyboard. Again, make sure that the Ø is not an O.

Another good thing to know about disks is how to keep them from being written on. The little notch on the inside of the disk sleeve can be covered with a tab or a piece of tape.

Then when the disk is in the disk drive, programs can be read from the disk into the computer, but no changes can be made to the programs on the disk. Any disk that you consider "finished" should be protected with a tab on that notch. If you change your mind, you can always pull it off again.



Cover notch with a tab or piece of tape

By now you are probably tired of all the different things you have to remember about the disk drive. It is really hard to memorize all the different commands.

Fortunately, there is a better way! At the very end of this book there is a program that you, your teacher, or one of your parents can type in to make the disk drive easier to use. It is called MENU, and after you've typed it in once, you can put a copy of it on every disk you have. Then, instead of having to remember the codes for how to format, erase, get a list of programs on a disk, or do several other handy things, you can simply choose them from the MENU. It will even make copies of itself automatically!

If the MENU program is too hard for you to type in, ask someone who knows more about computers to help you. It will save you from lots of work and problems in the long run.

SECTION 5: Getting Ready to Program

When you write a program, you are writing a list of instructions the computer needs to do a particular job, such as printing your name on the screen. These instructions are called **program statements**, and you'll learn more about them in Section 6.

But sometimes you need to tell the machine itself to do something, such as get rid of an old program so you can write a new one, or clear all the printing off the TV screen. These are called **commands**, because they are not part of a program. You type them in and press RETURN, and the computer does them right away.

(I wrote RETURN in a box, because it has a special key all its own on the keyboard, like CTRL and RESTORE.)

Program statements all have **line numbers** in front of them, to tell the computer which statement should be done first. (You will see these line numbers in Section 6.) Remember that commands do not have a line number because they are not part of a program.

Here are some of the **commands** you will need.

SHIFT — CLR HOME — this clears all the printing off the screen, but it does *not* take your program out of the memory. Remember—just because the information isn't printed on the TV screen doesn't mean it isn't stored inside the computer any more!

*Whenever the two keys are *connected* with a line, it means to hold down the first key while you press the second key.

NEW
RETURN — this erases your last program from the memory so you can start on a new program with a "clean" memory.

LIST
RETURN — this prints out, in order, whatever program statements you have typed into the memory so far.

If you type LIST, you will see all of the statements in your program. If you just want to see some of the statements:

```
LIST 20 <------- will LIST just line 20.
LIST 30-65 <---- will LIST lines 30 through 65.
```

RUN
RETURN

this tells the computer you have finished typing in all the instructions in your program, and now you want the computer to do that job. This is called executing the program. When the computer is finished executing the program, it will print READY and ■ on the screen. This box is called the cursor, and it will blink off and on to show where you will be typing next.

RUN
STOP

If the computer is in the middle of executing your program and you want it to stop, press RUN STOP. The computer will print

```
BREAK IN 10
```

which means that line 10 is the last line the computer worked on before you made it stop. For example, if the computer program was stopped at line 45, the computer would print

```
BREAK IN 45
```

CONT
RETURN

If you change your mind and want the computer to continue executing the program after you pushed RUN STOP, type in CONT and press RETURN. The computer will continue at the place it stopped.

A special key you will use often is the SHIFT key.

Many keys have two characters on the top of the key, such as % 5 or # 3 .

If you want to type what is shown on the bottom, just press the key.

If you want to type what is shown on the top, hold down the SHIFT key and press the key you want.

$ 4 ← to print this, hold down SHIFT and press the key.
← to print this, just press the key.

There are two SHIFT keys on the keyboard. You can use either one.

SECTION 6: PRINT and Variables

Let's begin by writing a program using the PRINT statement.

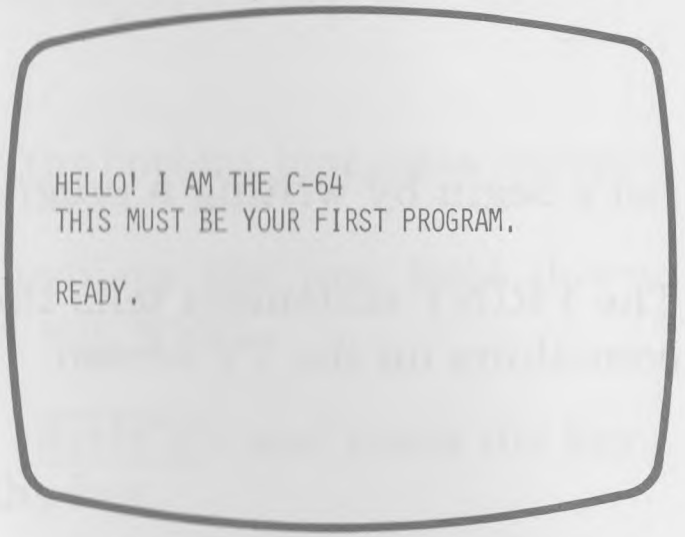
The PRINT statement tells the computer you want it to PRINT something on the TV screen:

```
10 PRINT "HELLO! I AM THE C-64"
15 PRINT "THIS MUST BE YOUR FIRST PROGRAM."
20 END
```

. . . and the last line in the program will be an END statement to show the computer where the program ends.

Now—when we type in RUN and press the RETURN key, this is what the computer will show on the screen:

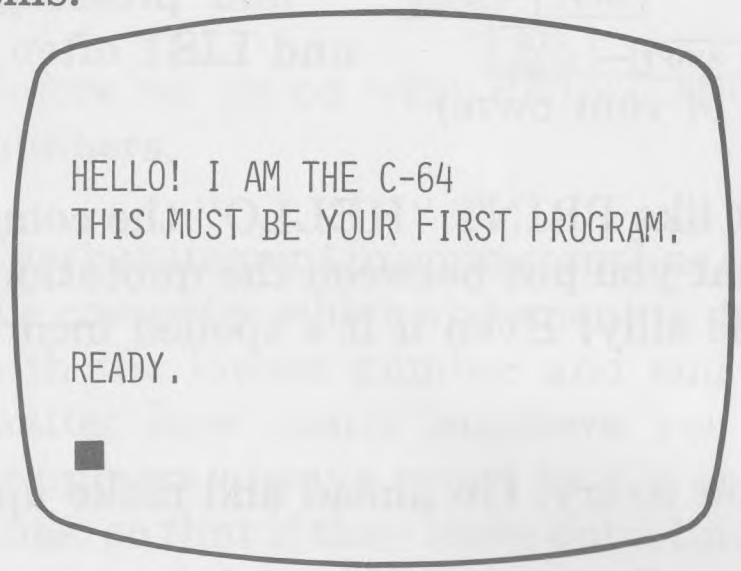
You try typing it in. Remember to press RETURN each time you finish typing a statement (before you begin with a new line number). RUN does not have a line number in front of it.



This is how the computer will follow the directions you gave it in your program.

When the computer is done following your instructions, this little square (called a cursor) will appear, which shows that you are ready to type again. The screen will also say "READY."

Suppose you want the computer to run the same kind of program, but this time you want the instructions erased from the screen before the program is executed. In other words, you want the computer to erase the screen first, before it follows your instructions. Then when you RUN your program, the screen would look like this:



To do this, all you need is one more statement in your program.

5 PRINT " [SHIFT] [CLR HOME] "

is a statement which tells the computer to erase the screen. Here's where it should go in your program:

```
 5 PRINT " [SHIFT] [CLR HOME] "
10 PRINT "HELLO! I AM THE C-64"
15 PRINT "THIS MUST BE YOUR FIRST PROGRAM."
20 END
```

When you actually type [SHIFT] [CLR HOME] in line 5, what you will see on the screen is ♥ (an inverse video heart). This is fine, don't worry. It will still work.

Print " SHIFT + CLR HOME " is a very handy statement to remember. It helps you get any "garbage" off the screen that might be in the way when you run your program.

(You can also use SHIFT + CLR HOME as a command, without a line number, just by typing SHIFT + CLR HOME and pressing RETURN. You will use SHIFT + CLR HOME and LIST often, when you write long programs of your own.)

Whenever you use a statement like PRINT "HELLO" the computer will print out **exactly** what you put between the quotation marks. Even if what you put is silly! Even if it's spelled incorrectly!

Here are some examples for you to try. Go ahead and make up some of your own!

```
1Ø PRINT " SHIFT + CLR HOME "
15 PRINT "I KIN SPELL REEL GOOD."
2Ø PRINT "GIGGLE!  GIGGLE!"
25 END
RUN
```

(Run is *not* part of the program, but I'm putting it here so you don't forget to type it in every time you want to *run* your program. Later, I won't write it down each time.)

```
2Ø PRINT " [SHIFT] [CLR HOME] "
3Ø PRINT "MY NAME IS JOHN SMITH."
4Ø PRINT "MY NAME IS MARY JONES."
5Ø END
RUN
```

```
1Ø PRINT " [SHIFT] [CLR HOME] "
15 PRINT "I AM A FRIENDLY COMPUTER."
2Ø END
RUN
```

Before we go on with PRINT statements, let's talk about line numbers.

Every statement in a program has a number in front of it. This tells the computer which statement to do first. The computer will start with the lowest number and end with the highest number, no matter how many numbers you skip in between. Good programmers always count by 5's or 1Ø's when they number their lines, so that if they leave out a line by mistake, they can put it in later, and there will be room. For example:

```
15 PRINT "MY NAME IS ROBBIE."
2Ø PRINT "MY BIRTHDAY IS JULY 3RD."
25 END
```

Now—if I wanted to put a line in my program telling how old Robbie is, right after line 15, I could just type in:

```
18 PRINT "I AM 9 YEARS OLD."
```

If I type SHIFT + CLR HOME to clear the screen, then type LIST, the computer will put line 18 into the program, in the right place, and this is how the program will appear:


```
15 PRINT "MY NAME IS ROBBIE."
18 PRINT "I AM 9 YEARS OLD."
20 PRINT "MY BIRTHDAY IS JULY 3RD."
25 END
```

This is very helpful if you forget something in your program.

You can use the same idea to **delete** (take out) a line in your program if you make a mistake or just decide you don't want that line anymore.

```
15 PRINT "TODAY IS TOOSDAY."
2Ø END
```

In this program, line 15 has a spelling mistake. To get rid of line 15 completely, all you do is type in:

```
15
```

and hit the RETURN key. This will erase line 15 from the memory, and you can type in a new line 15.

But, suppose you are typing a line and you notice right away that you've made a mistake, even before you go on to the next line. Can you erase part of a line? OF COURSE! All you have to do is press the CSRS key until you move the cursor on top of the mistake. Now you can type in the correct letters. If you want to move the cursor to the right, press the CRSR key. To move to the left, you use SHIFT + CRSR .

Remember that the CRSR key only works for the line you are typing on right then. If you are typing on line 15, you cannot erase something on line 5 with the CRSR key. You will have to type in a whole new line 5.*

*(You can read your C-64 manual to learn about fixing *any* line in your program.)

The INST DEL key is helpful, too. It lets you **delete** (or erase) what you have typed on a line. Just keep hitting the INST DEL key until you've erased what you want to get rid of.

There is one rather tricky thing to remember when you type long statements.

The C-64 will only print 40 characters on a line, so if you have a very long program statement like:

```
20 PRINT "MY NAME IS DAN GREENWOOD. I AM
THE WORLD'S BEST DONKEY KONG PLAYER."
```

your program will fill two lines on the screen before you are done typing. That is the longest that a single program statement can be. If you want a sentence to continue, you'll have to use another program statement. Remember to type RETURN before going on to a new statement (with a new line number). If you don't, the computer will think everything you've typed on the screen is one huge statement. When you try to RUN your program, it will be a big mess!

You may also be wondering why zero is written with a line through it, like this:

Ø

This is done on computers so that there is no mix-up between the number zero and the letter O. You should use the special zero when you write your programs on paper, too.

If you type something into the C-64 that it doesn't understand, it may print

```
?SYNTAX ERROR
```

on the screen. This means you have made a spelling mistake in a statement or command, or you have used the wrong statement. These messages from the computer are called **error messages**. They help you figure out what kind of mistake you have made, so you can fix it.

This computer has several different error messages, for different kinds of mistakes. The one you will see most often is

```
?SYNTAX ERROR IN 25
```

This means that the C-64 could not run your program because it found a mistake in line 25.

The PRINT statement can also be used to skip a line.

```
10 PRINT "[SHIFT][CLR HOME]"
15 PRINT "HELLO"
20 PRINT
25 PRINT "GOOD-BYE"
30 END
```

Without the quotation marks in a PRINT statement, the C-64 will work like a calculator:

```
10 PRINT "[SHIFT][CLR HOME]"
15 PRINT 10 + 20
20 END
```

This program will print out the answer to 10 + 20, which is 30. If you wanted the computer to print out the actual problem 10 + 20, you would write it like this:

```
10 PRINT "10 + 20"
```

Notice the difference?

Here is a PRINT program and the results on the screen when the program is executed. Look it over carefully.

```
10 PRINT "[SHIFT][CLR HOME]"
15 PRINT "RED"
20 PRINT "    BLUE"
25 PRINT "YELLOW"
30 PRINT "5 + 6"
35 PRINT 5 + 6
40 PRINT
45 PRINT "PINK PICKLES"
50 END
```

Notice that line 50 END does **not** print the word "END" on the screen. It just tells the computer that this is the **end** of your program.

The computer can also keep a number in its memory, and print it out later when you ask for it. Let's look at how the memory works.

The memory is like a big Post Office, with letters of the alphabet on each "mailbox." You put a number in a "mailbox" by using a LET statement.

A	B	C	D
5	7	2	Ø

```
45 LET A=5
5Ø LET B=7
55 LET C=2
6Ø LET D=Ø
```

Now whenever you use this statement in your program,

```
7Ø PRINT A
```

the computer will print out the number or **value** in the mailbox called A. Of course, if you want the computer to print out the value of A, you must make sure you put a number in mailbox A earlier in your program, or the computer will assume you wanted the value in mailbox A to be *zero*. This works for all the memory locations. If you do not put a number in a memory location, the computer will assume the value is zero.

In computer programs, the letter names you give to the "mailboxes" are called **variables.**

If you write a statement like

```
1Ø PRINT A+B
```

the computer will look in **A** to see what the value is, then find the value for **B**, and add them together and print out just the answer for you. Here is an example:

```
15 PRINT "[SHIFT][CLR HOME]"
25 LET A=6
35 LET B=4
45 PRINT A+B
55 END
```

MEMORY

A	B
6	4

Later in your program, if you want to change the number stored in mailbox A, you can use another LET statement. This will erase the old value for **A** and put in the new one.

The C-64 uses a few special symbols for arithmetic:

Addition +	3 plus 4 is written as 3 + 4
Subtraction —	5 minus 2 is written as 5 – 2
Multiplication*	6 times 8 is written as 6*8
Division /	6 divided by 2 is written as 6/2

You can use the computer in the **command mode** to do math problems for you.

The **command mode** means that the computer executes each line as soon as you press the RETURN key. You are using the command mode when you type in NEW or LIST or RUN when you write programs.

PRINT can also be used the same way:

notice—no line number!

The answer to the problem will be printed on the screen as soon as you press RETURN.

SECTION 7: GOTO and INPUT

PRINT statements alone don't make very exciting programs. This section has two new statements which make programming more fun!

Let's look at each one, then write some simple programs.

GOTO tells the computer to *go to* the line number listed, and do what it says there.

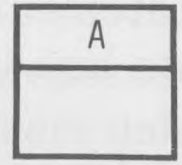
```
10 PRINT "HELLO"
15 GOTO 10
20 END
```

Every time the computer gets to line 15, the program tells it to go to line 10.

This program prints "HELLO" over and over and over again. The computer would print HELLO all night long, if you forgot to turn it off! Remember, you can stop the program by typing RUN STOP.

```
HELLO
HELLO
HELLO
HELLO
HELLO
```

INPUT asks you to type in a number while the program is running.




```
25 PRINT "TYPE IN YOUR AGE."
35 INPUT A
```

This sets up a memory space called A, and when you type in your age, it will be stored in memory space A.

Now we can use that information:

```
45 PRINT "YOUR AGE IS"
55 PRINT A
65 END
```

This line will print out the number stored in memory space A.

Type this program on the computer and try it yourself. You will notice that when the computer reaches an INPUT statement when it is running a program, it will stop and print "?" until you type in an answer. When you write your own INPUT programs, you must always be careful to put a statement before the INPUT telling the person who uses your program what the computer is waiting for them to type in.

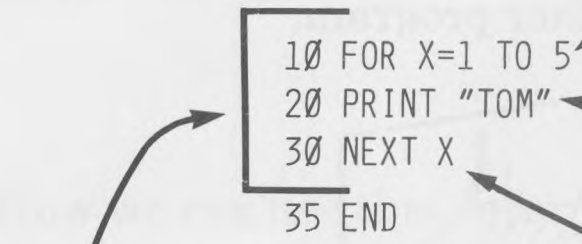
Sometimes you want to stop using a program with an INPUT statement in it, but you can't, because the computer keeps printing.

```
?REDO FROM START
```

no matter what you type in. To get the program to stop, type RUN STOP — RESTORE . Now you can LIST your program and fix the mistakes, or type NEW and write another program.

SECTION 8: IF-THEN and FOR-NEXT

FOR-NEXT statements are *lots* of fun, because you can make the computer do all kinds of work for you!



This statement says, "I'm going to do something 5 times."

What it will do is print "Tom". This statement tells it what to do each time.

This statement is the "counter." It counts how many times the computer has done its job. When it has done the job the right number of times, it will go on to the next line of directions.

This part of the program is called a **FOR-NEXT LOOP**, because the computer "loops" through that part of the program over and over again, until it has done its job the right number of times.

We can also write a program which has several lines between the FOR and NEXT statements in the loop:

FOR-NEXT LOOP

```
10 FOR X=1 TO 5
15 PRINT "MY NAME IS JIM LARSEN."
20 PRINT "I LIKE TO WRITE PROGRAMS."
25 PRINT "I HAVE MY OWN COMPUTER."
30 NEXT X
35 END
```

This program will write all three of the PRINT statements each time, until it has gone through the loop five times. It will print a total of 15 lines.

You may use any variable you wish in a FOR-NEXT loop, but the variable must be the *same* in both statements, or the computer will give you the error message NEXT WITHOUT FOR.

```
10 FOR Q=1 TO 12
15 PRINT "HARRY"
20 NEXT Q
```

these two variables must be the same

This program will print "HARRY" 12 times.

Let's look at how the counter works in a FOR-NEXT program.

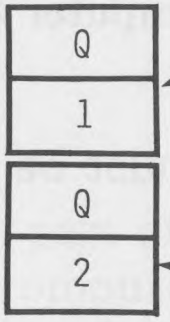
```
1Ø FOR Q=1 TO 4
15 PRINT "I LOVE COMPUTERS."
2Ø NEXT Q
```

This statement sets up a memory space named Q. It tells the computer that the **values** stored in Q will start with 1 and end with 4.

```
1Ø FOR Q=1 TO 4
15 PRINT "HELLO"
```

Each time the computer goes through the FOR-NEXT loop one time (and prints "HELLO"), the value of Q is increased by one.

After doing line 15 PRINT "HELLO" the first time, the number stored in Q is 1.



Now the computer goes back to line 1Ø to start the loop again. After it prints line 15 and gets to

```
2Ø NEXT Q
```

the number in Q is increased by 1. (that's what NEXT Q means.)

This goes on until Q finally gets up to 4. Since the FOR-NEXT loop says

```
FOR Q=1 TO 4
```

the computer knows that once Q gets to 4, the FOR-NEXT loop is finished, and the computer should go on to the next statement in the program.

Here are a few sample problems to try. Now take some time and write your own!

NAME

```
 8 PRINT " SHIFT CLR HOME "
1Ø FOR Z=1 TO 1ØØ
15 PRINT "SUSAN IS GREAT"
2Ø NEXT Z
3Ø END
```

NUMBERS

```
1Ø FOR R=1 TO 1ØØ
15 PRINT R
2Ø NEXT R
25 END
```

(In this program, you can see when the value of "R" changes!)

I have given these programs names, to make them easier to remember. Don't type in the name as part of the program, or the computer will give you an error message. (Of course, you could use these names when you save the programs on a cassette tape.)

IF-THEN statements provide a "test" for your programs.

```
30 PRINT "TYPE IN YOUR FAVORITE NUMBER."
40 INPUT N
50 IF N=5 THEN PRINT "YOU HAVE PICKED THE LUCKY NUMBER!"
60 END
```

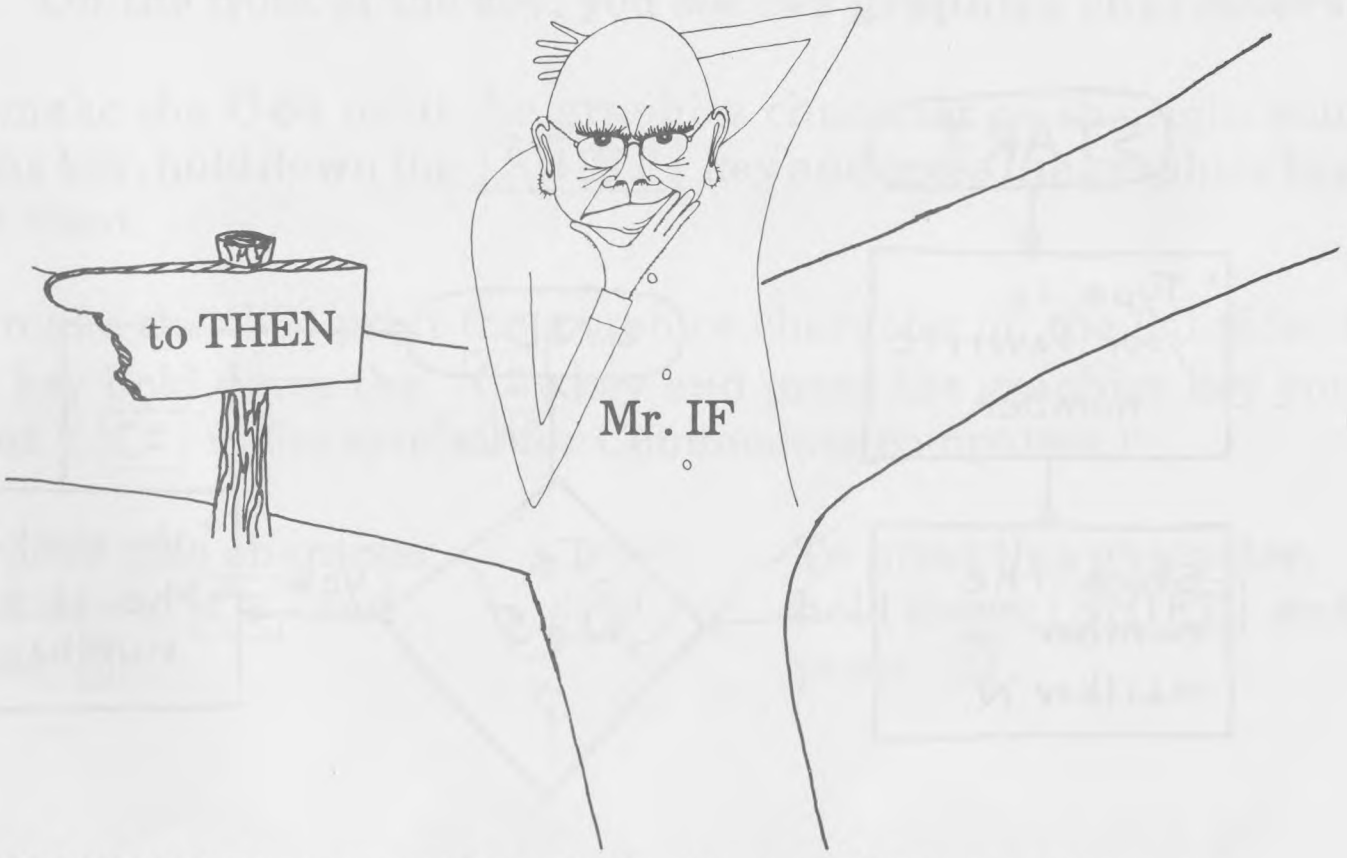
This statement looks in "mailbox" N to see what number is stored there in the memory. If it is 5, then the computer is told to PRINT "You have picked the lucky number!" If the number is *not* 5 (if the number "fails" the test), then the computer ignores the rest of the statement and goes on to the next line.

Let's think about how IF-THEN statements work.

Pretend you are "inside" your program, and you are following all the instructions in the program, just as the computer would.

You are going down the road, and you come to a fork, where there are two ways to go.

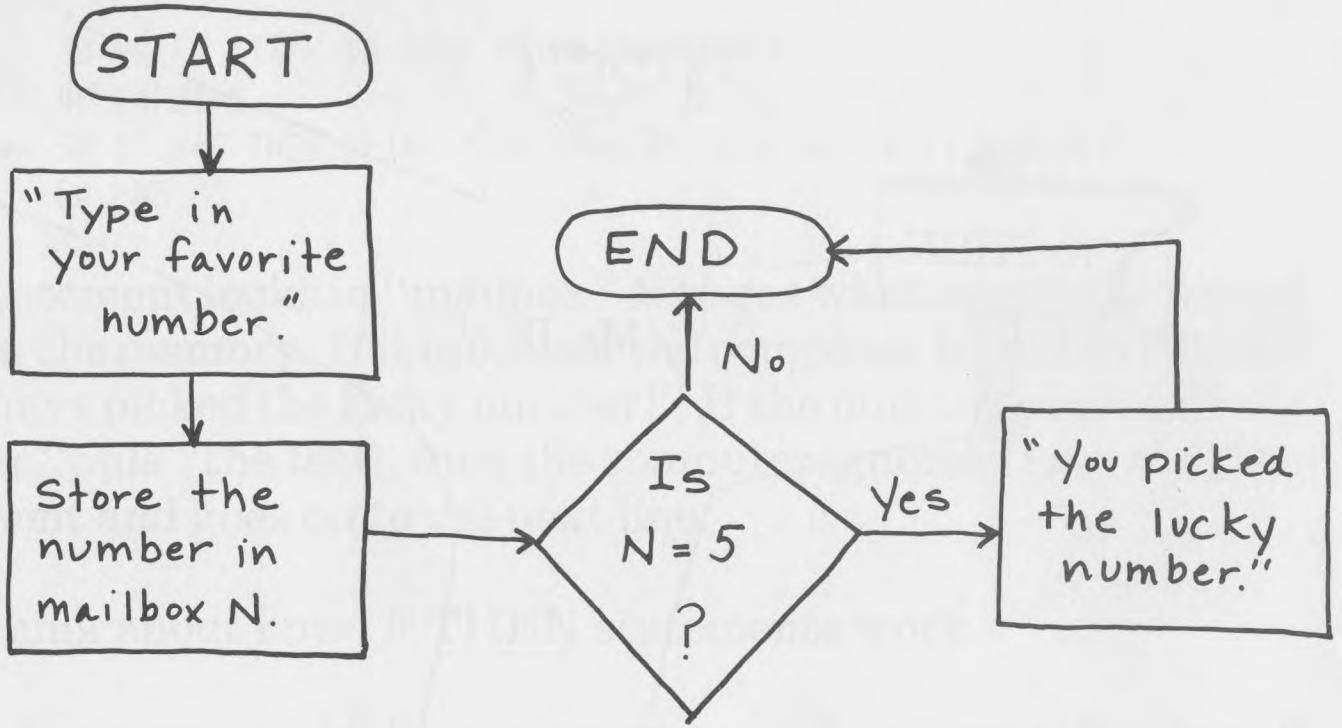
This is the IF-THEN statement in the program you are following. Mr. IF has a "test" for you. If you pass the test, you may go down the fork in the road marked "THEN." If you do not pass the test, you must go the other way.



An IF-THEN statement is called a **branch** in your program.

We can also show this with a flowchart.

```
30 PRINT "TYPE IN YOUR FAVORITE NUMBER."
40 INPUT N
50 IF N=5 THEN PRINT "YOU PICKED THE LUCKY NUMBER!"
60 END
```



SECTION 9: Graphics Programs

In this section we're going to learn to use all those funny graphics (pictures) you see on the keyboard. Most of the keys look like this:

They show a letter of the alphabet on top.

On the front of the key, you see two **graphics characters.**

To make the C-64 print the graphics character on the right side of the key, hold down the [SHIFT] key and press the graphics key you want.

To make the C-64 print the graphics character on the left side of the key hold down the [C=] key and press the graphics key you want. ([C=] is the symbol for Commodore computers.)

To print this character, hold down [C=] and press [Z].

To print this character, hold down [SHIFT] and press [Z].

You can combine all the different graphics characters to make pictures on the screen. Using the two cursor keys

[CRSR ↕] and [CRSR ↔]

lets you move the cursor anywhere on the screen. The cursor keys can be used with the [SHIFT] key to let you move up, down, left, or right.

[CRSR ↕]
- to move the cursor **up**, hold down [SHIFT] and press this key
- to move the cursor **down**, just press the key

[CRSR ↔]
- to move the cursor to the **left**, hold down [SHIFT] and press this key
- to move the cursor to the **right**, just press the key

Of course, the [INST DEL] still works when you are making pictures on the screen. Just press [INST DEL] to delete (erase) the last graphics character you printed. You can delete as many characters as you like.

It is easy to make pictures on the TV by using the graphics characters and moving the cursor around, so you can draw anywhere on the screen.

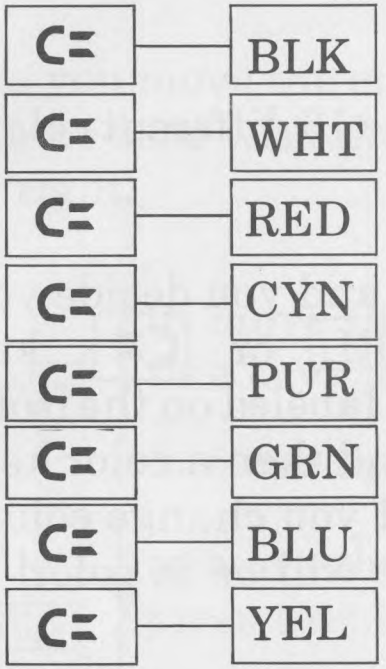
Now you will want to learn how to use all of the 16 different colors the C-64 can make.

When you are drawing pictures on the screen and you decide you want a new color, simply hold down the CTRL or C= key and press a color key. The first eight colors are labeled on the front of the number keys. By pressing CTRL and then a color key, everything you type will be in that color until you change colors again. Even the letters and numbers you type will be in color!

Here are what the three-letter names mean:

CTRL — BLK	Black
CTRL — WHT	White
CTRL — RED	Red
CTRL — CYN	Cyan (lighter Blue)
CTRL — PUR	Purple
CTRL — GRN	Green
CTRL — BLU	Blue
CTRL — YEL	Yellow

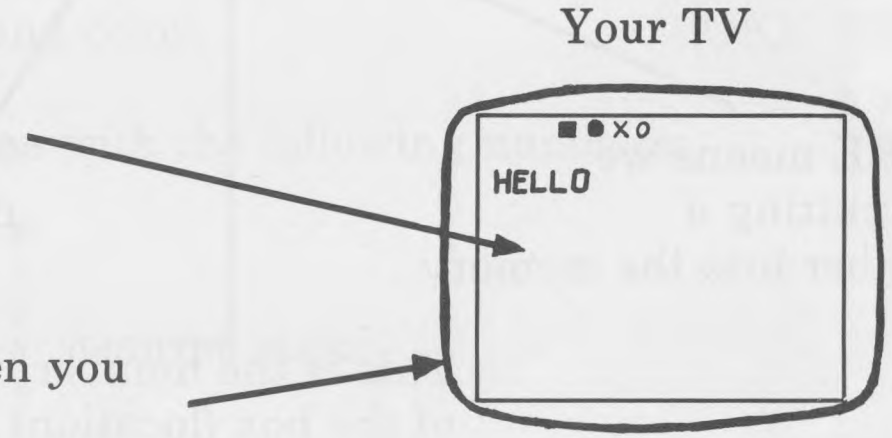
Another eight colors can be found by pressing the C= key along with a color key. Here are the colors to choose from:



The C-64 has an easy way to change the color of the whole screen and the border around the screen.

This is the background. When you first turn on your C-64, the background is BLUE.

This is the border. When you turn on your C-64, the border is CYAN (light blue).

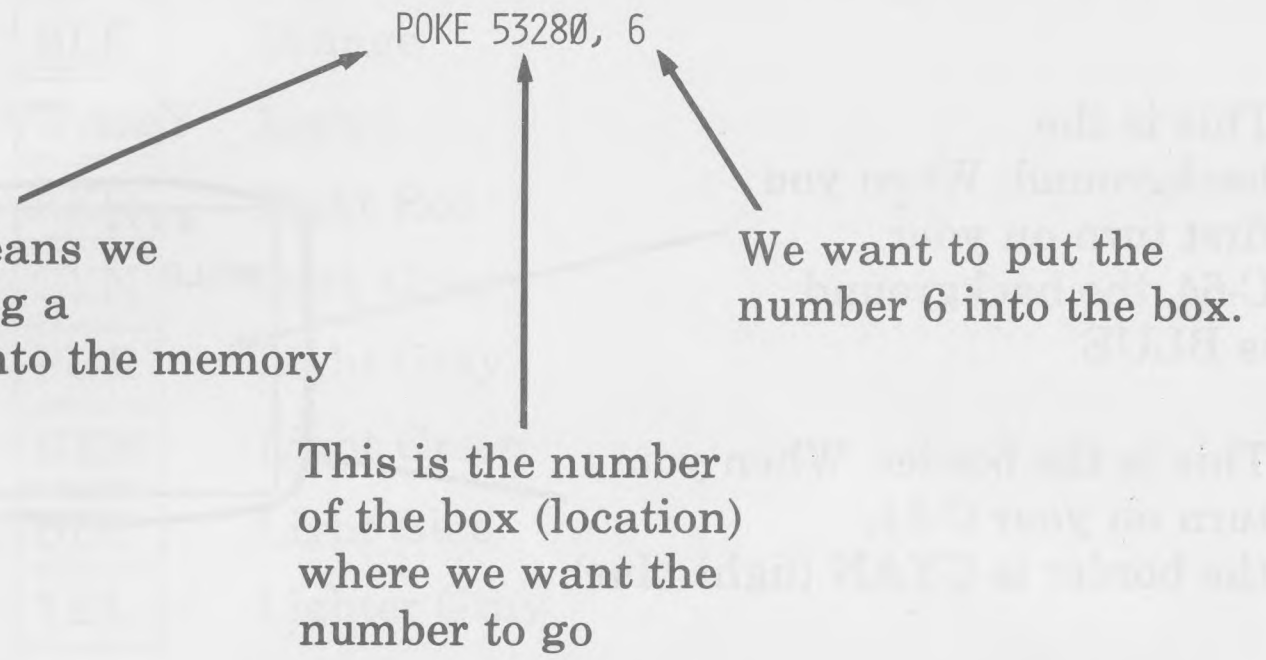


To change the border color and the background color, we use a special command called POKE.

POKE lets us put a number directly into the computer's memory.

In the memory, there are special "Post Office Boxes" or **memory locations,** which are identified by numbers. By putting numbers into special memory locations, or POKEing, we can get the C-64 to do what we want it to.

Location 5328Ø is one of these special places. By POKEing a number into it, we can change the border color of the screen.



Putting the number 6 into box number 5328Ø means something special to the C-64. It changes the border color from LIGHT BLUE to BLUE. After you type POKE 5328Ø,6 and press RETURN , it will look as though there is no more border on the screen. It is still there, though. The border is just the same color as the inner background.

To change the background colors, POKE location 53281. Try typing POKE 53281,1 RETURN . The background color will become WHITE.

Look at this table to find the background and border colors you want. Use their code numbers in POKE statements to change colors instantly!

To change border color: POKE 53280
To change background color: POKE 53281

POKE these locations with the following numbers:

BLACK	0	
WHITE	1	
RED	2	
CYAN	3	(LIGHTER BLUE)
PURPLE	4	
GREEN	5	
BLUE	6	
YELLOW	7	
ORANGE	8	
BROWN	9	
LIGHT RED	10	
DARK GRAY	11	
LIGHT GRAY	12	
LIGHT GREEN	13	
LIGHT BLUE	14	
LIGHTER GRAY	15	

For example, the following program will give you a green border and an orange background:

```
10 POKE 53280, 5
20 POKE 53281, 8
```

Sometimes the color combination will be really ugly or make the screen very hard to read. In this case, hold down [RUN STOP] while you press [RESTORE] . This will return the screen to the colors it usually uses.

Making colored pictures on the screen is lots of fun, BUT — what happens when you clear the screen, or turn off the computer? Your beautiful picture is gone forever!

There is another way to make graphics pictures on the C-64. If you put your graphics picture into a program, you can save it and run the program whenever you want to see the picture again. To make things easy to see, let's first make the background color white. Type POKE 53281,1 [RETURN] .

Let's say we want to make a very simple picture — a row of red hearts across the screen. We can use the PRINT statement to put this in a program.

```
10 PRINT
"[CTRL][RED][SHIFT][S][SHIFT][S][SHIFT][S][SHIFT][S]"
20 END
```

This will change the color to red, when we RUN the program

This will make the ♥ character, when we RUN the program

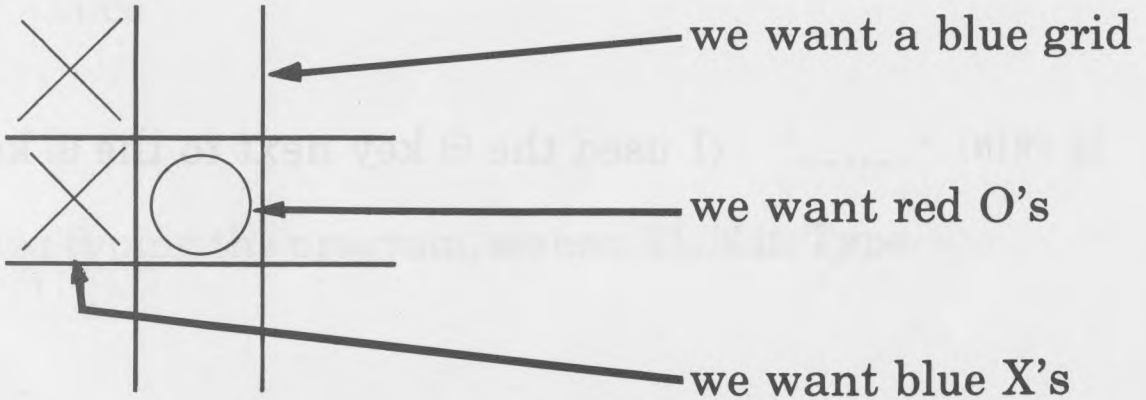
If you try to type this program on your computer, you will get a big surprise! When you hold down the [CTRL] key and press [RED], [£] appears on the screen instead!

The C-64 does this because it has to pick *something* to show on the screen, and it doesn't have a special character for pressing two keys at once. So it shows a regular character (like ◥ or ⇧) in **inverse video.** Inverse video means that the colors come out on the screen exactly opposite of what they usually are.

If you remember, the same kind of strange thing happens when you put `10 PRINT "` [SHIFT]—[CLR HOME] `"` in a program. The C-64 prints ♥ (inverse video heart) on the screen.

To make more interesting pictures, we can combine PRINT statements into a bigger program.

Let's write a program to draw a TIC-TAC-TOE game. We want it to look like this:



Our first statement should clear any old "garbage" off the screen:

```
10 PRINT "[SHIFT]-[CLR HOME]"
```

Now we need to write five PRINT statements:

```
20 PRINT "[CTRL]-[BLU][SHIFT]-[V][SHIFT]-[B]b[SHIFT]-[B]"
```

changes the color to blue

makes ⊠

makes ◫

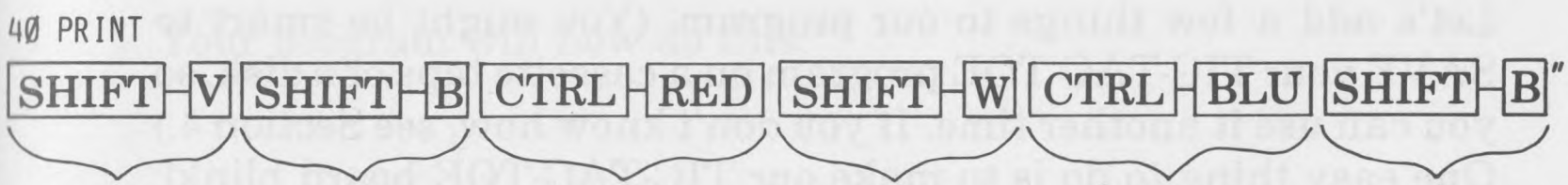
this means "blank" or hit the SPACE BAR once

makes ◫

```
30 PRINT "------"
```

(I used the ⊟ key next to the ⊞ key.)

```
40 PRINT
"SHIFT-V SHIFT-B CTRL-RED SHIFT-W CTRL-BLU SHIFT-B"
```



makes ⊠ (remember, our color is still BLUE)

makes ◫

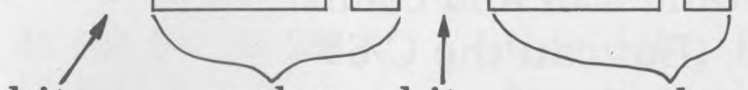
changes the color to RED

makes ◙

changes the color back to BLUE

makes ◫

```
50 PRINT "------"
60 PRINT "| b SHIFT-B b SHIFT-B"
```



hit SPACE BAR once

makes ◫

hit SPACE BAR once

makes ◫

```
70 END
```

When we are all finished typing the program, we can RUN it. Type RUN and press RETURN.

Let's add a few things to our program. (You might be smart to SAVE your TIC-TAC-TOE program on a cassette tape or a disk, so you can use it another time. If you don't know how, see Section 4.) One easy thing to do is to make our TIC-TAC-TOE board blink!

Add these four lines to your program:

```
15 FOR R=1 TO 5ØØ
18 NEXT R

65 FOR D=1 TO 5ØØ
68 NEXT D
```

These are special kinds of FOR-NEXT loops, called **delay loops**. They do what is between the FOR and NEXT statements (nothing!) 5ØØ times. This is like asking it to stand still and count to 5ØØ. (But can the C-64 ever count fast!!)

Now change line 7Ø so it says:

```
7Ø GOTO 1Ø
```

Your program will now do this:

```
1Ø
```
clears the screen

```
15 FOR R=1 TO 5ØØ
18 NEXT R
```
this delay loop leaves the screen blank long enough for your eyes to see it happen

```
2Ø
3Ø
4Ø
5Ø
6Ø
```
prints your TIC-TAC-TOE board

```
65 FOR D=1 TO 5ØØ
68 NEXT D
```
this delay loop leaves the TIC-TAC-TOE board on the screen long enough that your eyes can see it

```
7Ø GOTO 1Ø
```
this tells the computer to go to line 1Ø and start over again.

RUN the new version of your program and see how it works!

If you want to change the background and border colors for your TIC-TAC-TOE board, use POKE statements in your program.

```
5 POKE 53280, 4
6 POKE 53281, 5
```

These lines will give you a GREEN background and a PURPLE border. (WOW!)

Or, maybe you'd like to put a message in your program:

```
62 PRINT
```

will print a blank line

```
63 PRINT " CTRL-BLK ANYONE FOR TIC-TAC-TOE?"
```

changes the color to BLACK

here is what will be printed in black

You can make fantastic color pictures with your C-64. The first graphics picture you make is always the hardest to do — but after that, they're a snap!

When you start making big, complicated pictures, you might want to plan your pictures first on graph paper, using colored pencils or crayons. It helps you keep track of what you're doing.

SECTION 10: Sample Programs

COMPUTER PANIC

```
10 PRINT
15 PRINT "HELP!  THIS COMPUTER IS CRAZY!"
20 GOTO 10
```

GUESSING FUN

```
10 PRINT " [SHIFT]-[CLR HOME] "
15 PRINT "PICK A NUMBER.  TYPE IT IN."
20 PRINT "THE CHOICES ARE 1, 2, OR 3"
25 INPUT N
30 IF N=1 THEN PRINT "YOU WILL BE RICH."
35 IF N=2 THEN PRINT "YOU WILL BE FAMOUS."
40 IF N=3 THEN PRINT "YOU WILL HAVE 13 CHILDREN."
45 END
```

PINE TREE

```
5 PRINT " [SHIFT]-[CLR HOME] "
8 PRINT " [CTRL]-[GRN] "
10 PRINT "    X"
15 PRINT "   XXX"
20 PRINT "  XXXXX"
25 PRINT " XXXXXXX"
30 PRINT "XXXXXXXXX"
35 PRINT "XXXXXXXXXXX"
40 PRINT "    X"
45 PRINT "    X"
50 PRINT "    X"
55 END
```

make sure you skip enough spaces in each line!

ARITHMETIC

```
10 PRINT " [SHIFT]-[CLR HOME] "
15 LET A=1
20 LET B=2
25 LET C=3
30 LET D=4
35 PRINT "A+B="
40 PRINT A+B
45 PRINT
50 PRINT "C*D="
55 PRINT C*D
60 END
```

MR. MONSTER

```
10 PRINT "[SHIFT-CLR HOME]"
20 PRINT "MEET MR. MONSTER"
30 PRINT
40 PRINT
50 PRINT "[CTRL-BLK] [SHIFT-I] [SHIFT-I] [SHIFT-I]
          [SHIFT-U] [SHIFT-U] [SHIFT-U]"
60 PRINT
70 PRINT "[CTRL-RED] b [SHIFT-Z] bb [SHIFT-Z]"
80 PRINT "[CTRL-PUR] bb [SHIFT-Q] [SHIFT-Q]"
90 PRINT
100 PRINT "[CTRL-CYN] [SHIFT-J] ---- [SHIFT-K]"
110 END
```

here I used the ⊟ key next to the ⊞ key.

REMEMBER:

b means hit the SPACE BAR one time
(bb means two times)

Don't forget to press [RETURN] after you finish a statement! Just because the cursor jumped down to the next line doesn't mean you don't have to type [RETURN] at the end of a statement.

SECTION 11: Glossary of Statements and Commands

C= — the "Commodore Key" is used to get all the graphics characters on the left side of a key. Hold down the C= key and press the key you want.

CONT — continues to run a program after you stop the program by pressing RUN STOP .

CTRL — the control key lets you change the colors you are printing with. To choose a new color, hold down the CTRL key and press the color you want. This can also be used in a program statement: `10 PRINT "` CTRL + RED `"` (changes the color to red)

The CTRL key is also used to create special graphics characters or make special commands.

END — is a program statement which tells the computer the program is over: `95 END`

FOR-NEXT — a type of do-loop which has the computer perform some action a certain number of times.

Example:

```
10 FOR X=1 TO 100
15 PRINT "HELLO"
20 NEXT X
```

the computer will do this 100 times

GOTO—tells the computer to skip to a certain line number in the program.

```
1Ø PRINT "HELLO"
2Ø GOTO 1Ø
```

← this means go to line 1Ø and start over

IF-THEN—a type of branch statement which puts a "test" in the program. If the test is passed, the computer must follow special directions. If the test is not passed, the computer will drop down to the next line in the program.

```
1Ø IF N=5 THEN PRINT "THE NUMBER IS 5"
15 PRINT "WANT TO PLAY AGAIN?"
```

the computer will do this only if N = 5

INPUT — types out a question mark when the program is run, and waits for an answer to be typed in. The answer is stored in a certain memory space.

Example:
```
15 PRINT "TYPE IN YOUR AGE"
2Ø INPUT N
```

In this example, the answer is stored in memory space N.

LET—assigns a number to a memory space (or variable)

Example: `15 LET R=96`

You can leave the word "LET" out, and just type "R = 96", if you want to.

LIST — prints out a list, in order, of the program statements you have typed into memory. If you want to LIST just part of your program, you can type LIST 35-55, for example, to see lines 35 through 55.

LOAD — loads a program from a cassette tape or disk into the computer's memory. To load a program from a cassette type LOAD "MY PROGRAM". To load a program from a disk, type LOAD "MY PROGRAM", 8. Use the name of the program you really want to load in place of "MY PROGRAM".

NEW — erases the old program from memory.

POKE — tells the computer to put a given value into a special memory location. Can be used to change border and background colors.

PRINT — tells the computer you want to write something on the screen.

```
2Ø PRINT "MY NAME IS RALPH"
```

RETURN — you must press this key each time you finish typing a statement. This is how you tell the computer you are finished with that statement, and will be starting a new line number. (Just having the cursor jump to a new line is not enough.)

RUN — starts the execution of the program. This command is not part of the program itself, and does not have a line number.

RUN STOP — pressing this key stops the program while it is running. The C-64 will give you a message to say which line the program was on when it stopped. (BREAK IN 9Ø means it stopped at line 9Ø.)

RUN STOP + RESET ——— is handy for those times when a program is doing crazy things, and you just can't figure out what to do. If you press RUN STOP + RESET, the program will stop, but the memory will not be erased. Then you can try again.

SAVE — records your program on a cassette tape or disk. (This does not erase your program from the memory. Only NEW, or turning off the keyboard, will do that.) To save a program on a cassette, type SAVE "MY PROGRAM". To save a program on a disk, type SAVE "MY PROGRAM", 8. Type in the name of the program you really want to save instead of "MY PROGRAM".

SHIFT — pressing SHIFT and a graphics key lets you print the graphics character on the *right* side of the key. If a key has two symbols on the top of it, using the shift key will let you print the *top* symbol.

SHIFT + CLR HOME ——— clears off the screen so you can start printing again at the top.

VERIFY — "double checks" a program you have saved on a cassette tape or disk against the program in the computer's memory. If something went wrong, it will tell you. For cassette, type: VERIFY "MY PROGRAM". For disk, type: VERIFY "MY PROGRAM", 8. The name of the program you want to verify goes in quotes.

Notes for Teachers and Parents

I am not, by any stretch of the imagination, a sophisticated programmer. I took one programming course in college, and have spent the past four years working with elementary school children. I've taught microcomputer programming to nearly 3ØØ children, ranging in age from 4 to 12, and have yet to run into any children who aren't dying to get their hands on the keyboard. Computers are a *natural*—they give immediate feedback, and allow children to create something all their own. I love teaching programming. It is one of the most exciting things I've ever done.

Candidates for teaching programming to young children must have one trait above all others—the ability to interact with the children on a peer level, and learn along *with* them. Computer programming is not just a skill—it is a *tool.* You learn programming not as a study in itself, but because of what you can accomplish with it. In any one computer program, there are many ways to approach and solve the problem at hand. The thing I say most often to my students is, "Run it, and see if it works!"

I won't presume to tell you just how to go about teaching your particular group of children, but I would like to share some of my successful ideas, and some of my failures. These are the things which have worked, or not worked, with every group of children I've taught in the past few years.

One word from someone who's been there: book some computer time for yourself during each week, before you start teaching the children. Once they've had a few lessons, they'll insist you stand in line like everyone else!

General Hints

If you have had no previous experience with the C-64, before you do anything else, read through the manual that comes with the machine. (At the time this book went to press, several manuals, some better than others, were available for the novice programmer.) It is especially important to understand the directions for setting up the machine.

To gain an overview of this book, you may wish to read through the children's portion before you work through the C-64 manual. It will not be necessary to learn everything in the manual — check the glossary of this book for the most crucial statements and commands.

Obviously, this book cannot cover every function of the C-64, nor would it be appropriate to try to teach all of them to young children.

The Commodore 64 is capable of sophisticated animated graphics and sound synthesis. The concepts and methods involved in accessing these capabilities from BASIC are based in large part on cryptic POKE commands. They are difficult for even an adult to master, let alone a child. For that reason I have elected to omit those topics from this book.

Commodore has announced a ROM cartridge and other companies are now releasing software to make these capabilities easier to work with. I suggest that until such tools become available the topics of Sprite animation and SID sound should be left out of the child's computer curriculum.

It seemed to me that PRINT graphics provided more fun and success for the students in a shorter amount of time. Also, proofreading (and typing!) programs filled with large numbers is frustrating and time-consuming for the "hunt and peck" typist.

SETTING UP YOUR COMPUTER CENTER: Since programmers tend to get excited and vocal, I suggest that you locate your computer in a semi-secluded area which is near someone in your school who understands the machine. If the children have any problems with the computer, they are going to come and find you anyway, and it's easier if they don't have to call you down the hall from another room.

Secure the electrical cords in a way that keeps them out of the traffic pattern around the computer. Stepping on the cords may cause a fire hazard and will create fuzz on the TV screen. Remember that the power adaptor unit must be unplugged when not in use (such as at night).

I schedule only two children at the computer during one time slot. They usually help each other if they encounter difficulties. More than two children at one time may encourage fighting over who will do the typing.

Have enough room for several chairs around the keyboard, and consider where you will place the computer when you teach a group. Sometimes I have used a kitchen timer to keep the children moving, and other times I have run the schedule by the clock. It depends on your group. You will keep your sanity longer if you enforce the rule: "When your turn is over, it's OVER."

A computer notebook for each child is a must. They should learn to take notes on how to do things, or they will never become confident programmers. Discourage them from running to you for answers they should have in their notes.

You may wonder why I have so few sample programs in this book. I have found that the more timid programmers will never pull away from the safety of typing in *my* programs every time they are on the machine, unless I provide very few samples, and force them to think up their own.

It sounds, from the tone of these hints, that I have many problems with children who program. That isn't the case at all. However, a seemingly trivial problem can eat up precious time when 50 children are waiting to use one computer.

The most important thing you as the teacher must do is to give the children an *overall* view of the problem you are trying to program. They must see that a problem can be broken down into sections, and then each section can be accomplished on the computer in several different ways. Teaching only *what* a statement does, without focusing on *why* you would want to use it, creates frustrations for most children.

TEACHERS . . .

My biggest failure of all time. . . I can't emphasize this one enough. Don't let your class of students play commercial game tapes until they are accomplished programmers! By "accomplished," I mean the end of the first year for most children.

Let's face it—playing a computerized game is much more fun, and a lot less work, than learning to program. Especially for elementary school children. If they discover that they can play game tapes on the school computer, they will lose all interest in doing the work involved in learning to program. This is a sad but true fact, and one I learned the *hard* way. Even with your 12-year-olds, you will regret the day you ever brought a game tape into your computer center. Overnight, they will change from being thrilled about having the chance to try their own programs, to being disappointed because their favorite game tape has been retired. Certainly, they'll have a chance to play games on the computer. But the games should be those that *they* have written themselves.

GROUP INSTRUCTION: Choose for your computer center a room which has effective shades on the windows. When I teach a group, I place the TV facing them, and I sit at an angle to the keyboard. We talk over what we want to accomplish, and I do the typing. Unless you have a crackerjack typist in the group, it makes lessons unbearably slow if the whole class has to wait while someone hunts and pecks on the keys.

Suggested Lesson Outline — Once a Week Lessons

Do not go on to the next topic until all the children have had a chance to try out their last lesson on the computer, or they will never remember it. A weekly schedule is imperative for assuring individual children their time at the computer. With the exception of SAVE and LOAD, these lessons follow the sequence of the book.

SECTION 1

1. What is a computer?

SECTION 2

2. Introduce flowcharting
3. Practice writing flowcharts
4. More practice on flowcharts

SECTION 3

5. Using the keyboard, running the machine itself, behavior guidelines, scheduling, RETURN

SECTION 5

6. Beginning programming: SHIFT — CLR HOME, RUN STOP, CONT, NEW, LIST, RUN with PRINT examples

SECTION 6

7. PRINT statements with quotation marks, SYNTAX ERROR, SHIFT + CLR HOME in programs
8. PRINT to skip lines; editing
9. PRINT with arithmetic operations (+, −, *, /)
10. PRINT variables—simple
11. PRINT variables such as PRINT A + B

SECTION 4

12. Operation of the cassette recorder and/or disk drive (SAVE, LOAD, VERIFY)

SECTION 7

13. GOTO
14. INPUT

SECTION 8

15. FOR-NEXT with PRINT and arithmetic statements.
16. More work on FOR-NEXT
17. IF-THEN test
18. IF-THEN in more complex programs (draw flowchart of program function)

SECTION 9

19. Use of SHIFT, C=, cursor movement keys
20. Changing colors, POKE for background/ border colors
21. Graphics programs in one color
22. Graphics programs in many colors
23. Graphics used within other programs

SECTION 1Ø

24. Discuss possibilities for designing and carrying out their own programs

Teaching Suggestions For Each Section

Section 1: What is a computer?

Comparisons to home computer games are helpful. Most children think computers are "smart," and younger children will think they are "magic." Don't overexplain — make arrangements to get the children on the machine as soon as feasible. None of your explanations will really make sense until then. You might want to write a simple INPUT program they can use for some "hands-on" practice.

Section 2: Flowcharting

The objective is logical thinking, not perfection. Have fun! Choose 'How To' type topics, which have built-in choices. They must be about something the children understand.

How to: Give the dog a bath; Make pizza; Build a doghouse; Lose your allowance; Make a phone call to Grandma.

Set a minimum number of do-loops or branches. I usually set a minimum of three. Use manilla drawing paper. Have the children write the words first, *then* draw the boxes around the words. It's easier!

Section 3: Running the machine itself

Be careful to use the words "save" and "load" properly, when you talk about the cassette recorder, or the children will confuse the two terms. Typing practice on school typewriters or those at home will help the children accomplish much more during their turn at the computer. Children who practice on manual typewriters tend to pound on the computer keys. This will cause typing errors and other keyboard problems.

Section 4: Saving your programs with a cassette recorder

The cassette player system for saving programs is a headache for many people, but it is my opinion that they have trouble because they do one of these things incorrectly:

a. They use poor quality recording tape, or try to save programs over old recordings.
b. They don't leave enough space on the tape between programs.
c. They don't teach the children how to use the counter properly. (If you have one.)
d. They don't save one program twice on the same tape, to insure that at least one of them will turn out.

Actually, I have had fewer problems with the C-64 cassette system than with any other I have used. The VERIFY feature is really a time-saver.

Section 5: Getting ready to program

Written quizzes on the meaning of the commands and statements accomplish very little. Let the children learn about this while they type in their own programs. If they don't know what they're doing, their program simply won't run.

Teach the difference between *commands* and *statements.* If they do not understand this difference, the children will be frustrated the first few times they work at the computer. Typically, they will type in a program without any line numbers, and then will not be able to understand why none of the program is in the memory to be listed later.

Section 6: PRINT and variables

At the back of the book, you will find samples of PRINT worksheets. This is one of the few areas in which worksheets are of real value.

Discourage the children from using the letter 'O' as a variable. It tends to be confusing.

Section 7: GOTO and INPUT

Every PRINT line moves the cursor down to the next line. This problem will come up when you teach FOR-NEXT loops with several PRINT statements in the middle. Look up the use of the semi-colon in the C-64 manuals. Sooner or later, someone will ask you how to make things print right next to each other on the same line when using several PRINT statements.

Section 8: IF-THEN and FOR-NEXT

Reminder: variables must match on FOR-NEXT loops.

The children will have trouble remembering that the computer drops down to the next line if the test is not met in an IF-THEN statement.

Flowcharts may be helpful in tracing the functions of loops and branches in more complex programs. Don't be overly concerned with detail when you draw them.

'X is not equal to Y' is written as
X < > Y or X > < Y.

Tracing the "Numbers" program through its memory changes is helpful for showing how the values of variables change in a program. (This program may be found at the end of Section 8.)

Section 9: Graphics programs

The screen is 40 characters wide by 25 lines long. Remember that the "READY" message after a program has finished takes up a line, too.

Have the children plan their pictures on graph paper.

When the children number the squares of their graph paper, have them put an 'X' in the upper left-hand corner square. This helps prevent mistakes in numbering.

X	1	2	3	4	5	6	7
1							
2							
3							
4							

Use crayons or markers to plan graphics pictures in the actual colors you will use.

Section 10: Sample programs

As stated earlier, with the more timid programmers, I find that too many examples inhibit experimentation.

If you are using a cassette recorder, all the children should have their own cassette for saving their programs. Buying some gummed labels is a good investment, so that they will not have trouble trying to locate those programs at a later date.

Section 11: Glossary

For those examples in which I used statements, as opposed to commands, I put line numbers in front of each program line, to help remind the children.

MENU: The Easy Way to Use the C-64 Disk Drive

In the disk portion of Section 4, a barrage of disk do's and don'ts are presented, along with a set of command codes that can easily lead to frustration. Unfortunately, the Commodore disk drive is a bit more complex than need be, and even something as simple as obtaining a disk directory must be done in several steps. Even more unfortunately, the manual that comes with the disk drive is difficult to decipher.

As an alternative to using the command codes, I strongly recommend typing in the program that follows and saving it to disk. It makes the commands outlined in Section 4 much easier to grasp, and performs them automatically. It also adds a few new capabilities that make disk management much easier.

WARNING! When you load the MENU program into memory, you will *erase* any other program you had in the memory. Don't forget to SAVE the program in the memory before you LOAD the MENU program from the disk, or you will lose your program.

When you run the program, you are presented with a menu. You may select any function simply by pressing a number on the keyboard, then RETURN . Here is a description of each function:

1. DISK DIRECTORY. Pressing 1 RETURN displays the disk name, number, and a complete directory of the current disk. The blocks occupied by each program are also indicated along with a readout on file type and the number of blocks remaining. On a disk with many files, press CTRL to slow the list, and RUN STOP to stop it.

2. FORMAT NEW DISK. By pressing 2 RETURN , you can format a disk automatically. The program will ask you to insert the disk you wish to format, then to type in the disk name and number you've chosen. It will then format the disk under the given name and number. Remember to hit RETURN after putting in the file information.

3. INITIALIZE DRIVE. Pressing 3 RETURN will reinitialize the drive. Earlier Commodore disk drives required reinitialization every time a new disk was inserted. Newer drives need only be reinitialized to recover from an error condition (see condition 9). Think of this as a way to turn the disk drive off and back on without having to turn off the computer.

4. COPY FILE. An existing file can be copied under a new file name by pressing 4 RETURN . In this way you can keep an unaltered copy of a file and create a copy to modify. The program will ask you to input the name of the source file (original program), and the name you wish to give the copy. Don't forget to hit RETURN after answering the questions. The program will then write the copy to disk under the given file name.

5. RENAME FILE. You can change the name of an existing file by pressing 5 RETURN . The program will ask you to input the old file name, then the name to which you wish to change it. Hit RETURN after each response. The program will then change the file name for you.

6. ERASE (SCRATCH) FILE. You can delete files from the disk by pressing 6 RETURN . The program will ask you for the file name to delete, then to confirm the command. It will then delete the file.

7. VALIDATE FILES. When a disk has been written to repeatedly, bits and pieces of free space are wasted. By pressing 7 RETURN , the VALIDATE command will do a bit of housekeeping on the disk, and reclaim lost blocks of memory here and there. In this way you can get the most out of your disk space.

8. WRITE MENU FILE. This automatically writes the MENU program itself to disk. When starting a new disk, first load the menu program from an old disk. Insert the blank, then use the format selection (2) to format the new disk. Then press 8 [RETURN] to write the menu program itself to the new disk. Your new disk will then contain the MENU utility.

9. ERROR STATUS. If the disk encounters an error during any MENU function, pressing 9 [RETURN] will return the specific error, along with a track and sector location if applicable. The error query will automatically reset the drive to non-error status as well, so that you can try the original operation again. A read out of ERROR Ø means no error at all.

10. EXIT TO BASIC. By pressing 1Ø [RETURN], you will exit the MENU program. Remember to type NEW [RETURN] before entering a new program. MENU will remind you to do this as it signs off.

Control Characters Used in This Listing

Character	Description	Function	How to Get It On the Keyboard	Used in Program Line(s)
♥	Inverse heart	Clear screen and home cursor	SHIFT + CLR HOME	10, 250, 300, 350, 400, 450, 500, 550, 600, 670, 700
£	Inverse British "pound" sign	Red	CTRL + RED	262
R	Inverse R	Turns inverse on	CTRL + 9	262, 276
═	Inverse underline	Turns inverse off	CTRL + 0	262
←	Inverse left arrow	Blue	CTRL + BLU	262, 282
↑	Inverse up arrow	Green	CTRL + GRN	276
▌	Medium block left of center	Black	CTRL + BLK	10
▐	Inverse vertical (centered)	Cursor left	SHIFT + ←CRSR→	276

```
5 POKE 53280,1:POKE 53281,1
10 PRINT"■□"
20 PRINT" VIC-1541 USER'S MENU"
30 PRINT"-------------------------"
40 PRINT
50 PRINT"   1. DISK DIRECTORY"
60 PRINT"   2. FORMAT NEW DISK"
70 PRINT"   3. INITIALIZE DRIVE"
80 PRINT"   4. COPY FILE"
90 PRINT"   5. RENAME FILE"
100 PRINT"   6. ERASE (SCRATCH) FILE"
110 PRINT"   7. VALIDATE FILES"
120 PRINT"   8. WRITE MENU FILE"
130 PRINT"   9. ERROR STATUS"
140 PRINT"  10. EXIT TO BASIC"
150 PRINT
160 PRINT"-------------------------"
170 PRINT" INPUT NUMBER OF YOUR":PRINT"
CHOICE, HIT <RETURN>
180 PRINT
190 INPUT CHOICE
200 ON CHOICE GOSUB 250,300,350,400,450,
500,550,600,650,700
210 GOTO 10
250 PRINT"□"
251 OPEN 1,8,0,"$"
252 GET #1,A$,B$
254 GET #1,A$,B$
256 GET #1,A$,B$
258 C=0:IF A$<>""THEN C=ASC(A$)
260 IF B$<>""THEN C=C+ASC(B$)*256
262 PRINT"■■"MID$(STR$(C),2);TAB(5);"■■
  ";
264 GET #1,B$:IF ST<>0THEN 282
266 IF B$<>CHR$(34)THEN 264
268 GET #1,B$:IF B$<>CHR$(34) THEN PRINT
B$;:GOTO268
270 GET #1,B$:IF B$=CHR$(32) THEN 270
272 PRINTTAB(25);:C$=""
274 C$=C$+B$:GET #1,B$:IF B$<>""THEN 274
276 PRINT"■■"LEFT$(C$,3)
280 IF ST=0 THEN 254
282 PRINT" BLOCKS FREE■"
```

```
284 CLOSE 1:PRINT:PRINT:PRINT"HIT <RETUR
N> FOR MENU":INPUT X$:RETURN
300 PRINT"□":PRINT:PRINT
305 PRINT"INSERT DISK TO BE":PRINT"FORMA
TTED.":PRINT
310 PRINT"INPUT DISK NAME" :INPUT DISK$
320 PRINT:PRINT "INPUT DISK NUMBER":INPU
T EXT$
325 MACRO$="N:"+DISK$+","+EXT$
330 OPEN 15,8,15,MACRO$
340 CLOSE 15:MACRO$="":RETURN
350 PRINT"□":PRINT:PRINT
380 OPEN 15,8,15,"I"
390 CLOSE 15:RETURN
400 PRINT"□":PRINT:PRINT
410 PRINT"INPUT SOURCE FILE NAME" :INPUT
 DISK$
420 PRINT:PRINT "INPUT NEW FILE NAME":IN
PUT NWS$
425 MACRO$="C:"+NWS$+"="+DISK$
430 OPEN 15,8,15,MACRO$
440 CLOSE 15:MACRO$="":RETURN
450 PRINT"□":PRINT:PRINT
460 PRINT"INPUT OLD FILE NAME" :INPUT DI
SK$
470 PRINT:PRINT "INPUT NEW FILE NAME":IN
PUT NWS$
475 MACRO$="R:"+NWS$+"="+DISK$
480 OPEN 15,8,15,MACRO$
490 CLOSE 15:MACRO$="":RETURN
500 PRINT"□":PRINT:PRINT
510 PRINT"INPUT FILE NAME TO":PRINT"DELE
TE":INPUT DISK$
520 PRINT:PRINT:PRINT"HIT <RETURN> TO DE
LETE":INPUT X$
530 MACRO$="S:"+DISK$
535 OPEN 15,8,15,MACRO$
540 CLOSE 15:MACRO$="":RETURN
```

```
550 PRINT"□":PRINT:PRINT
560 PRINT"WARNING:OPEN AND RANDOM ACCESS
 FILES ":PRINT"WILL BE DELETED"
570 PRINT:PRINT:PRINT"HIT <RETURN> TO":P
RINT"VALIDATE":INPUT X$
580 OPEN 1,8,15,"V"
590 CLOSE 1:RETURN
600 PRINT"□":PRINT:PRINT
610 PRINT"INSERT DISK TO BE":PRINT"WRITT
EN TO.":PRINT
620 PRINT:PRINT:PRINT"HIT <RETURN> TO WR
ITE":PRINT"MENU FILE":INPUT X$
625 OPEN 1,8,15
630 SAVE "MENU",8
635 CLOSE 1
640 RETURN
650 OPEN 1,8,15
660 INPUT#1,A,B$,C,D
670 PRINT"□":PRINT:PRINT
680 PRINT"ERROR STATUS":PRINT:PRINT"ERRO
R # ";A:PRINT B$:PRINT"TRACK ";C,"SECTOR
 ";D
690 PRINT:PRINT:PRINT"HIT <RETURN> FOR M
ENU":INPUT X$:CLOSE 1:RETURN
700 PRINT"□":PRINT:PRINT
710 PRINT"NOTE:  MENU PROGRAM IS":PRINT"S
TILL RESIDENT."
715 PRINT:PRINT"TYPE <NEW> TO CLEAR MEMO
RY."
720 END
```

NAME _____

Simulate these computer runs. Show your "printout" on the screen.

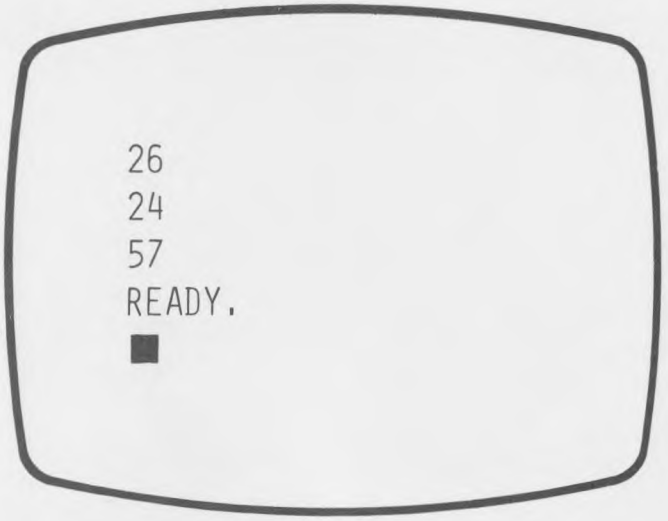
```
10 PRINT "[SHIFT][CLR HOME]"
20 PRINT "BIG"
30 PRINT
40 PRINT "YELLOW"
50 PRINT "BLUE"
60 END
```

```
10 PRINT "[SHIFT][CLR HOME]"
20 PRINT "THE ANSWER"
30 PRINT 30 * 2
40 PRINT 30 + 2
50 PRINT 30 - 2
60 PRINT "THE END"
70 END
```

NAME _____

Here is a program and "printout." Find and fix the mistakes in the program so a run will produce what is shown on the screen.

```
10 PRINT "[SHIFT]-[CLR HOME]"
20 PRINT 20 + 6
30 PRINT 30 + 4
35 PRINT
40 PRINT "60 - 3"
50 PRINT 10 - 10
60 PRINT "HELLO"
70 END
```



$5.95

For the child who is eager to enter the exciting world of computers
and
for the parent or teacher who is eager to help them learn

Computers for Kids is here!!

The Computers for Kids series is designed for children ages 8-13 who are interested in computers but are overwhelmed by the reading level and quantity of material covered in adult level manuals. (Computers for Kids can be considered a children's level manual.)

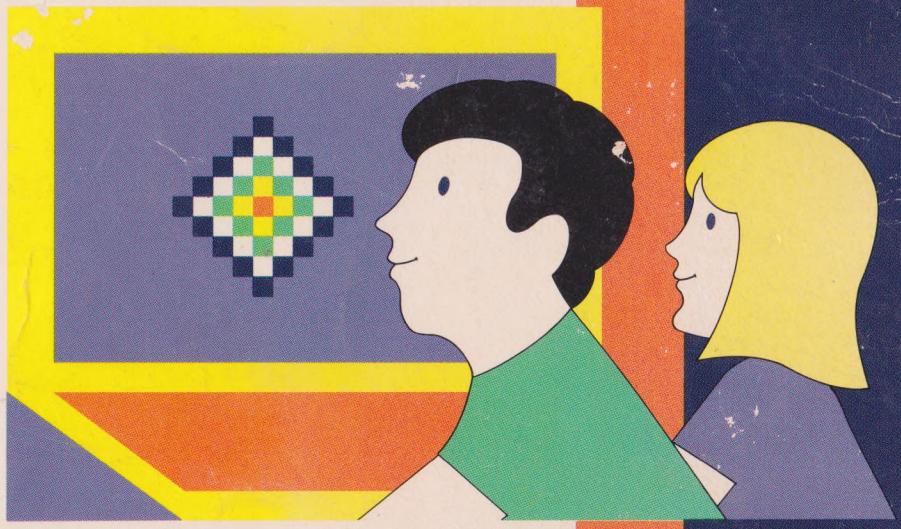
This entertaining, easy to read book includes sections on:

- How computers work
- How to operate the Commodore 64
- Simplified flowcharting
- Step by step discussion of BASIC
- Statements and how they are used to write programs
- A child's glossary of BASIC terms
- Color Graphics

Sally Greenwood Larsen has taught Kindergarten through Seventh grade and is a pioneer in teaching young children to program microcomputers.

She has also written editions in this series for the TRS-80, Sinclair, Atari, Commodore VIC-20, IBM PC and Apple II Plus computers.

**There is a special section for parents and teachers with teaching ideas, trouble-shooting hints, lesson plans.



CREATIVE COMPUTING PRESS

ISBN 0-916688-63-1