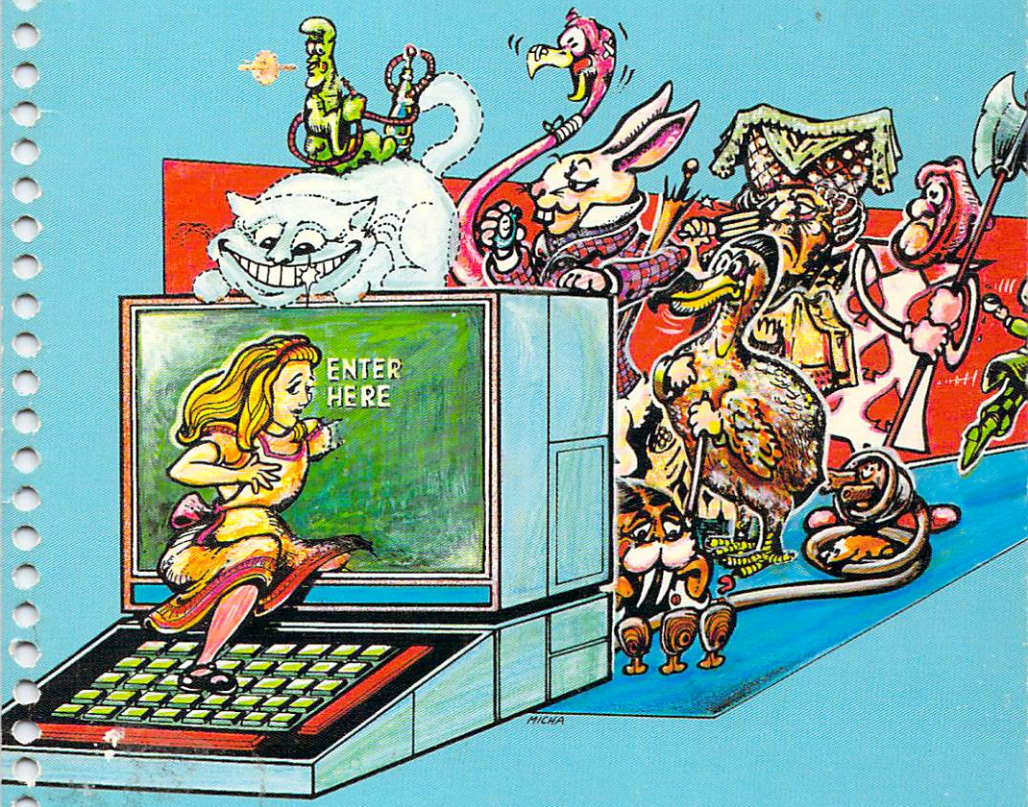
# COMMODORE 64® PUZZLEMENTS?

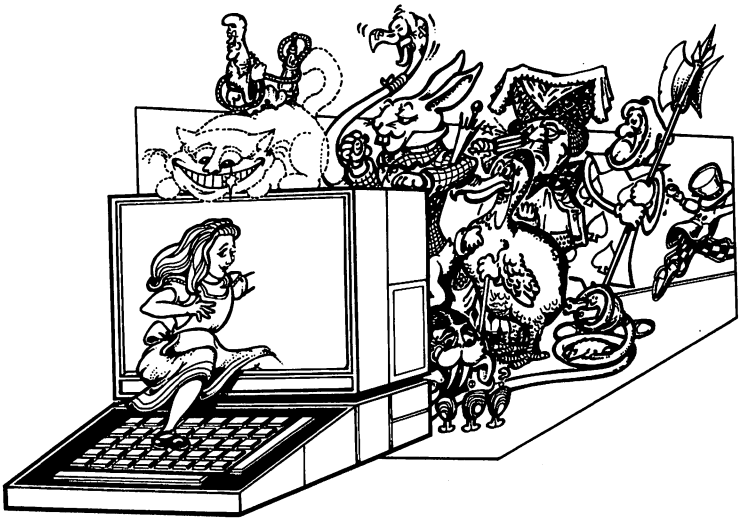
## HERBERT KOHL




A CREATIVE PASTIMES BOOK

This book belongs to

# COMMODORE 64® PUZZLEMENTS?



by HERBERT KOHL

illustrations by C. MICHA

A Creative Pastimes Book
Reston Publishing Company
A Prentice-Hall Company
Reston, Virginia

**The author wants to thank Jim Bach for his work on translating the programs in this book into Commodore 64 BASIC.**

**Library of Congress Cataloging in Publication Data**

Kohl, Herbert R.
Commodore 64 puzzlements.

"A Creative Pastimes book."
1. Commodore 64 (Computer) 2. Computer games.
I. Title.
QA768.C64K64 1984 795.8'2 84-3490
ISBN 0-8359-0788-1



10 9 8 7 6 5 4 3 2 1

Printed in the United States of America.

# Contents



# Introduction

Most computer puzzles are versions of ordinary word games, such as crossword puzzles or anagrams, that draw their vocabulary from the world of computing. The puzzles here are different. They provide games, amusements, and challenges from within the world of computing itself and require some familiarity with Commodore BASIC. The puzzles in this first volume are quite simple, though we plan additional volumes with more complex and sophisticated puzzles. You do not have to be a computer buff or a skillful programmer to solve them. In fact, they would be an ideal complement to any text or reference manual for a person of any age who is just learning BASIC and is beginning to feel the power of creating programs.

We have included very few PEEKs or POKEs in the puzzles; however, each time we use one it is fully explained. Also the programs that are the bases of the puzzles are no longer than 15 lines. These puzzles are designed to help you think in BASIC, learn to read a program, and understand something about program structure. All the commands in the book are included in the **Commodore BASIC Reference Manual** or in any introductory text on BASIC. Since the programs are not that long or complex (though some are tricky and not easy to solve automatically), many can be done with a paper and pencil. In fact, it's a good idea to try to think through many of these problems without using your computer. Program design is a mental activity that is computer-assisted and working on computer problems without using a computer can help develop that skill. In order to help you sketch out solutions, we've provided some formats designed to help with problem solving. They are in the Appendix to the book and you should feel free to copy them.

There are five different types of problems in the book: Error Statement Problems, Line Scrambles, Line Number Scrambles, Missing Code Lines, and Control Character Graphics Design Problems. Some of the programs deal with numbers; others deal with words. Color and graphics are often used. The programs themselves have been selected to illustrate certain aspects of Commodore BASIC such as nested loops, subroutines, branching, and the mix of graphics, words, and numbers. You might even find it useful to incorporate some of the smaller programs here as parts of larger programs you build yourself.

The answers to the programs will provide explanations of what the program does and give hints that might help you solve other problems. They will also give you a correct program. **Remember, however, that**

**often there is more than one correct answer and, if you feel correct, run your program. If it works you have come up with another solution!!**

Here is a simple example that should give you a sense of the specific nature of the different types of puzzles in the book:

```
10 PRINT "ON A SCALE OF 1 TO 10"
20 PRINT "HOW DO YOU FEEL TODAY?"
30 PRINT "1 IS MISERABLE, 10 GREAT"
40 INPUT X
50 IF X>5 THEN PRINT "HAVE FUN!":END
60 PRINT "MAYBE THIS WILL HELP.."
70 PRINT "THINGS ARE NEVER AS BAD"
80 PRINT "AS THEY SEEM."
90 PRINT "TAKE A DEEP BREATH. COUNT"
100 PRINT "TO 10 AND SLOWLY SMILE"
```

This program asks how you feel on a scale of 1 to 10, 1 being miserable and 10 great. If you answer with a number above 5 the computer prints "HAVE FUN!" and the program ends. If you answer 5 or less and indicate you are not feeling too great, the computer gives you some friendly advice on how to cope with being down.

This program can be used to illustrate the nature of different puzzle types in this book.

## Error Code Puzzles

In these puzzles, there will be an error for you to figure out. It could be on line 50, for example, which would then read:

```
50 errorIFX > 5:THEN PRINT"HAVE FUN"END
```

The error would be the colon between 5 and THEN.
Here's another possible error. Can you figure it out?

```
30 error PRINT "1 IS MISERABLE,10 GREAT
```

You probably guessed that the error was the missing quotation marks at the end of the line.

## Line Scrambles

In these puzzles, one line is all scrambled up. You have to unscramble it to make the program work. Here's a scramble of line 50:

```
50 PRINT 5 THEN > "X" END FUN IF :HAVE
```

Throughout the book there will either be descriptions of how the program will run or pictures of what should appear on the screen. These are called **screen dumps**. If you leaf through the book you'll see lots of them.

## Line Number Scrambles

Instead of just scrambling a line, these puzzles mix up all the line numbers. You have to renumber each line to put the program in order and have it run as planned. This requires some thought and experimentation and it is here that you are likely to find more than one correct unscrambling of a program.

A very simple scramble would reverse all of the line numbers so that

line 10 becomes line 100
line 20 becomes line 90
line 30 becomes line 80
etc.

However, you are not likely to come upon such patterned scrambles. The line number patterns do not provide hints to the unscrambling of the lines. You have to think through to the program structure to solve the puzzle.

## Missing Code Lines

In this variation on program puzzles, one line of code is missing. It might be in our sample:

```
40 PRINT "?????????????"
                        or
80 PRINT "?????????????"
```

PRINT "?????????????" is the indicator of the missing code line. That does not mean that PRINT,", or ? necessarily appear on the missing line.

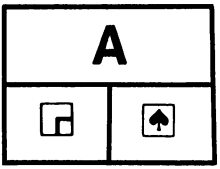
## Shift Character Graphics Design Problems

In addition to letters, your Commodore 64 has graphics characters on the keyboard. You can get this set by holding down the shift key and pressing any of the other keys, or by holding down the Commodore key and doing the same things. Notice that each key on the Commodore keyboard has two graphics characters printed on it. To get the character on the left, you press:

C= [KEY]

To get the character on the right you press:

SHIFT [KEY]



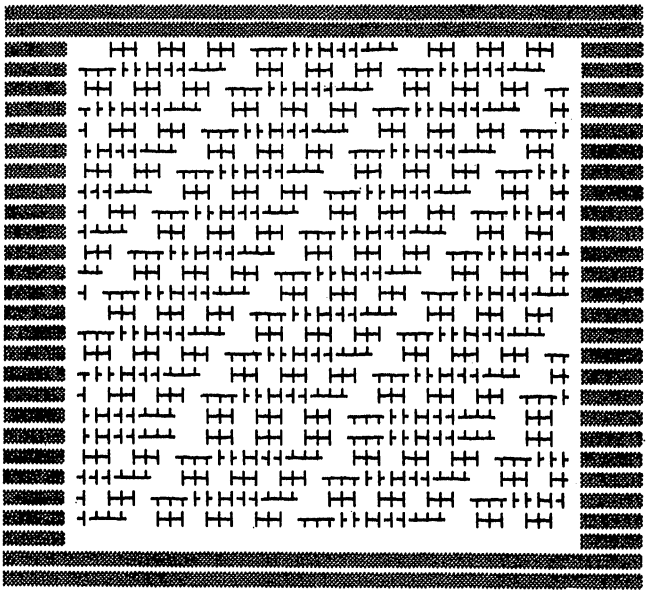

Thus for the A key

C= [A] gives you:

and

SHIFT [A] gives you:

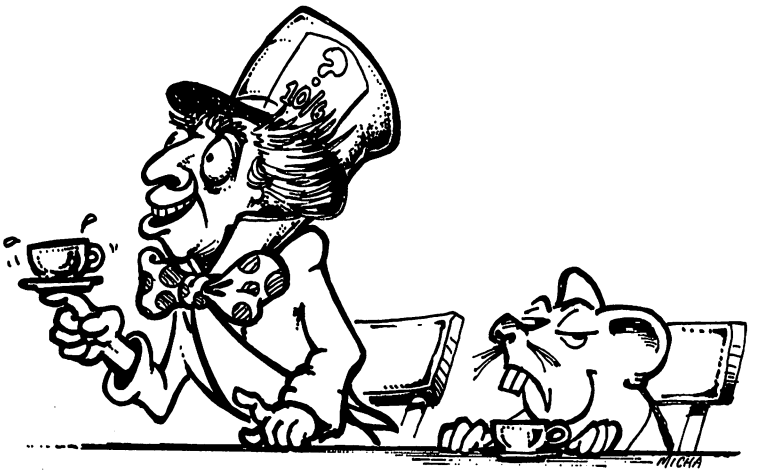
These characters can be combined to make interesting patterns and drawings. They can also be used within your program. Here's a short program that produces a fancy design using control character graphics. The screen dump shows you how the program looks when it is running.



```
10 PRINT"┌┐└┘╋━┃ ";
20 PRINT"  ╋  ╋  ╋ ";
30 GOTO10

READY.
```

A number of interesting puzzles can be created using character graphics. A simple one would provide you with a screen dump and ask you to create a program to produce that pattern. There are a number of other challenges in this section as well.

Now that you have a sense of the kinds of challenges in this book try your hand at solving them. Each section begins simply and then moves on to more complex demands. The answers and their explanations are at the end of each chapter. I hope you have as much fun solving the puzzles as I had inventing them.




# Error Puzzles

## or getting your program to do what you plan

One of the most frustrating things about learning to program a computer is that you can work hard at a program and make a tiny mistake that throws the whole thing off. It is particularly annoying if you have not internalized the programming language you are using. This first section contains some puzzles that embody the simplest mistakes everybody makes when learning to program. These mistakes should be looked at as puzzles to solve rather than as signs of your inability to master computing.

# 1

# A SIMPLE CRYSTAL BALL

This program asks whether you like the number 3 or 4 better. When run, this is how it is supposed to respond:

```
RUN
WHICH NUMBER IS YOUR FAVORITE
OF THESE TWO: 3 OR 4
? 3
YOU HAVE MYSTICAL POWERS

READY.
RUN
WHICH NUMBER IS YOUR FAVORITE
OF THESE TWO: 3 OR 4
? 4
YOU ARE DEEP AND INTUITIVE

READY.
■
```

However, your program simply bombs when you input the number 3. It seems ok if you input 4:

```
RUN
WHICH NUMBER IS YOUR FAVORITE
OF THESE TWO: 3 OR 4
? 3

READY.
RUN
WHICH NUMBER IS YOUR FAVORITE
OF THESE TWO: 3 OR 4
? 4
YOU ARE DEEP AND INTUITIVE

READY.
■
```

Here is the program. What is the mistake?

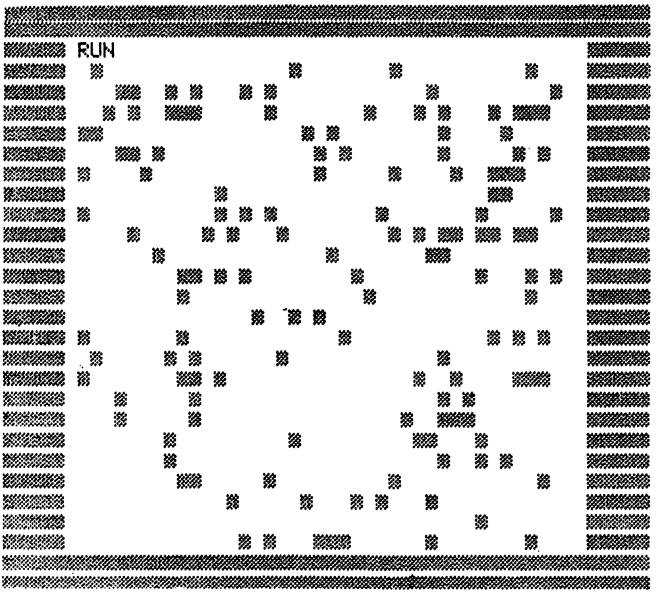
```
10 PRINT"WHICH NUMBER IS YOUR FAVORITE"
20 PRINT"OF THESE TWO: 3 OR 4"
30 INPUTX
40 IFX=3THENGOTO1010
50 IFX=4THENGOTO1500
1000 PRINT"YOU HAVE MYSTICAL POWERS"
1010 END
1500 PRINT"YOU ARE DEEP AND INTUITIVE"
1510 END

READY.
```

# 2

# FILL IT UP

In this program the intent was to fill the screen up with a random distribution of characters.

Instead you got this on the screen:




How could you change this program to give you the full screen?

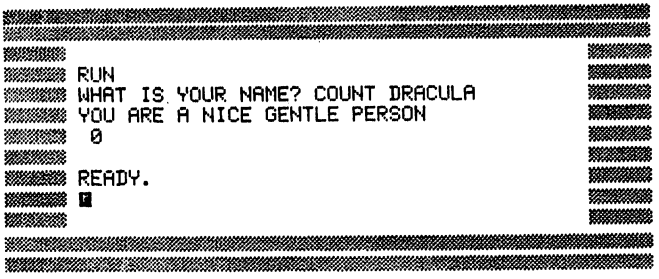
```
10 PRINTCHR$(19);
20 Y=RND(1)*12+1
30 FORZ=1TOY:PRINT:NEXT
40 PRINTTAB(RND(1)*39)"%";
50 GOTO10

READY.
```

RUN

# 3

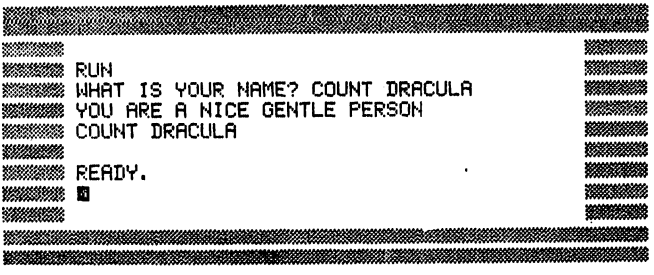
# COUNT DRACULA

You write a simple program asking your friend's name. You want the computer to tell your friend that he or she is a nice person. One of your friends decides to play around and type in Count Dracula and here is what happens with your program:




```
10 INPUT"WHAT IS YOUR NAME";A$
20 PRINT"YOU ARE A NICE GENTLE PERSON"
30 PRINTA

READY.
```

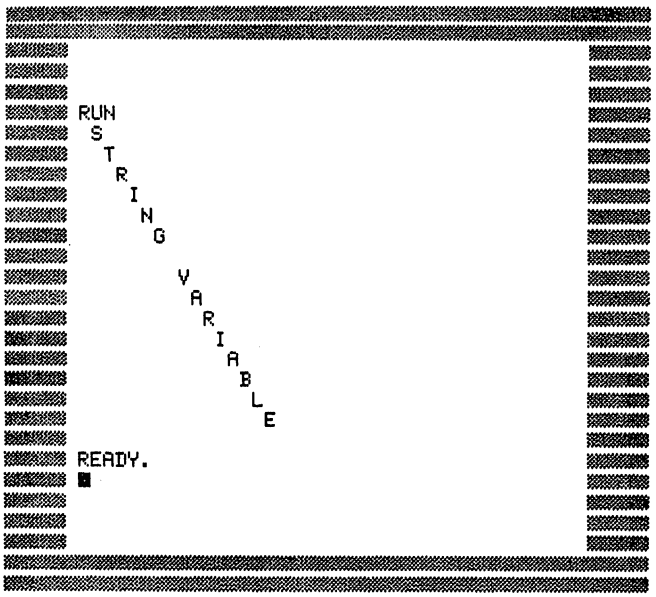
How can you fix your program so that it gives Dracula his full compliments like this?



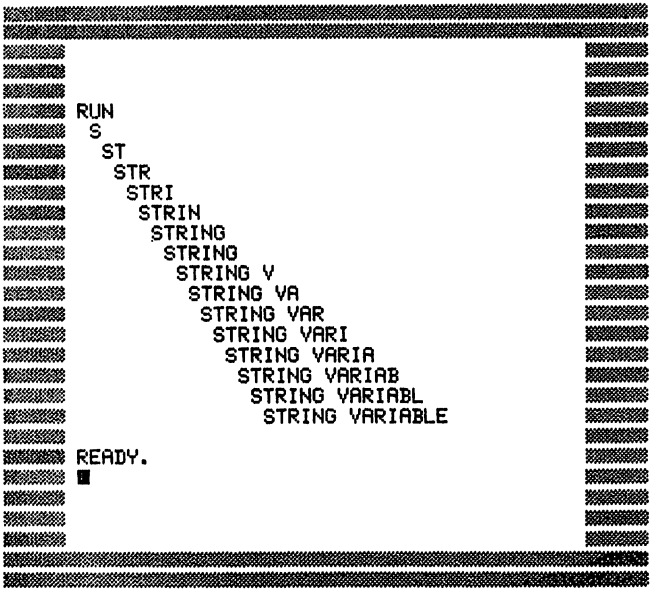
# 4

# STRING THE VARIABLES ON

You want to print out STRING VARIABLE running diagonally down the screen like this:




However, what you get is this:



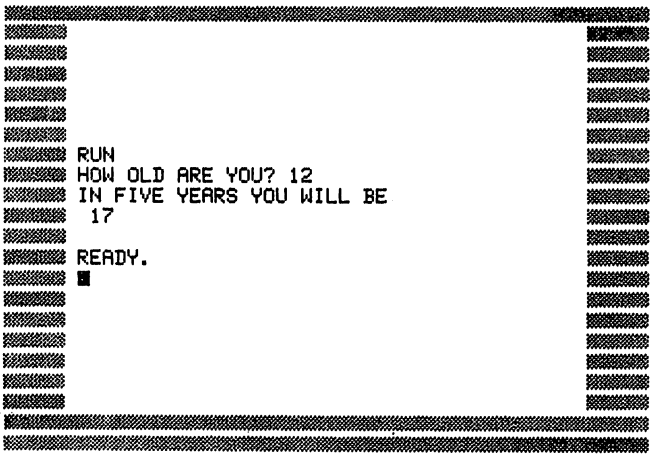

Here's your program. What went wrong?

```
10 A$="STRING VARIABLE"
20 FORX=1TO15
30 PRINTTAB(X)MID$(A$,1,X)
40 NEXT

READY.
```

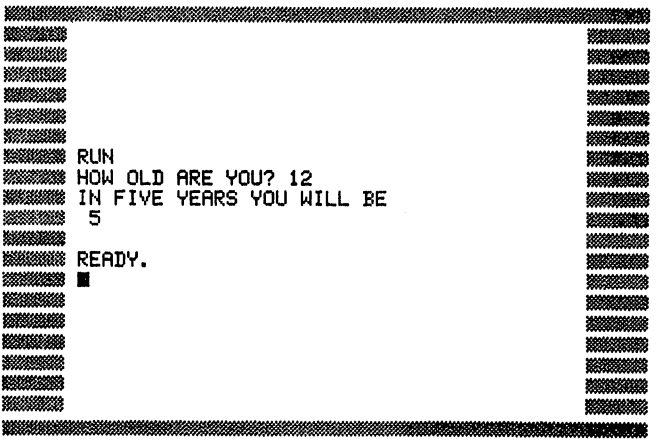
# 5

# IN FIVE YEARS

This simple program is supposed to ask a person's age and then tell them how old they will be in 5 years. It should run like this:




Instead it runs like this:



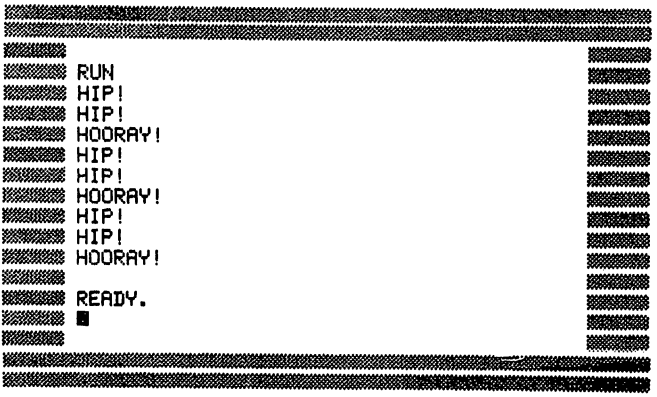
Here's the program. Where's the error?

```
10 INPUT"HOW OLD ARE YOU";A
20 PRINT"IN FIVE YEARS YOU WILL BE"
30 PRINTX+5

READY.
```

# 6

# HIP, HIP, HOORAY!

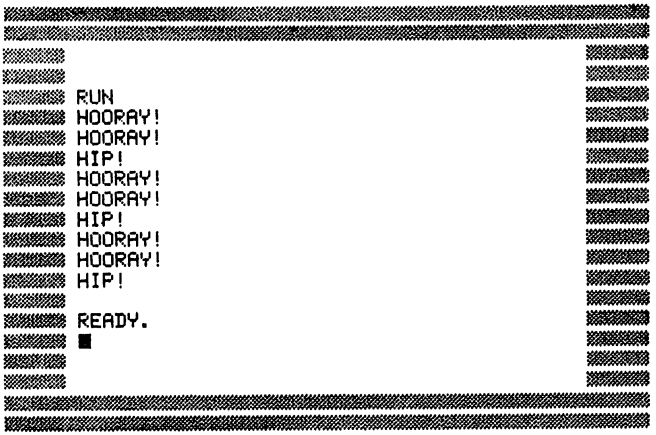
This is a reverse puzzle. Here is a program that will print HIP! HIP! HOORAY! three times as in the screen dump:



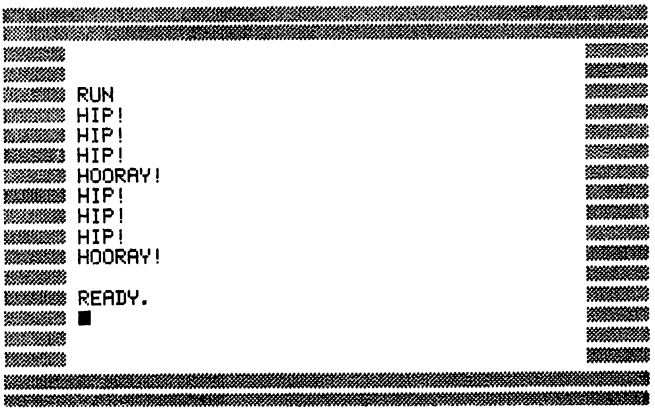

```
10 FORA=1TO3
20 FORB=1TO2
30 PRINT"HIP!"
40 NEXTB
50 PRINT"HOORAY!"
60 NEXTA

READY.
```

Now here are three similiar screen dumps that almost do what the program does. Can you construct three programs that lead to these screen dumps?

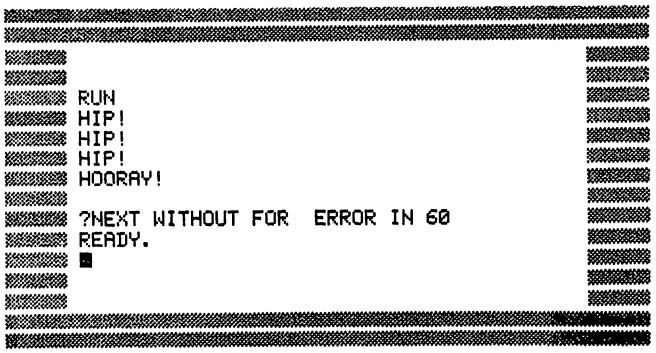
A


RUN
HOORAY!
HOORAY!
HIP!
HOORAY!
HOORAY!
HIP!
HOORAY!
HOORAY!
HIP!
READY.

B


RUN
HIP!
HIP!
HIP!
HOORAY!
HIP!
HIP!
HIP!
HOORAY!
READY.

C


RUN
HIP!
HIP!
HIP!
HOORAY!
?NEXT WITHOUT FOR  ERROR IN 60
READY.

# Answers

## 1

At line 40 you instructed the program to GOTO line 1010 which ended the program. You always have to be careful that you have your program jump to exactly the lines you want it to. It makes sense to check all GOTO statements if you have a program that doesn't run properly. Here's the program with the intended reference:

```
10 PRINT"WHICH NUMBER IS YOUR FAVORITE"
20 PRINT"OF THESE TWO: 3 OR 4"
30 INPUTX
40 IFX=3THENGOTO1000
50 IFX=4THENGOTO1500
1000 PRINT"YOU HAVE MYSTICAL POWERS"
1010 END
1500 PRINT"YOU ARE DEEP AND INTUITIVE"
1510 END

READY.
```

## 2

The problem was at line 20. You only used half the screen. You have to be careful about the screen dimensions. At line 20 the number 12 should be 24 as in this program:

```
10 PRINTCHR$(19);
20 Y=RND(1)*24+1
30 FORZ=1TOY:PRINT:NEXT
40 PRINTTAB(RND(1)*39)"▒";
50 GOTO10

READY.
```

## 3

Here is a program that will give Dracula his due:

```
10 INPUT"WHAT IS YOUR NAME";A$
20 PRINT"YOU ARE A NICE GENTLE PERSON"
30 PRINTA$

READY.
```

Notice that at line 10 there is an A$ indicating that there will be an alphabetic input. However, on line 30 the $ has been dropped so that you get a 0 returned instead of a word. It is important to check that letter variables are properly referenced by including the $ after the variable name.

## 4

Here is a program that will give you what you wanted:

```
10 A$="STRING VARIABLE"
20 FORX=1TO15
30 PRINTTAB(X)MID$(A$,X,1)
40 NEXT

READY.
```

Notice that at line 30 the midstring command MID$(A,B,C) had its variables reversed. Instead of MID$(A$,X,1), which gives you what you want, the error program had MID$(A$,1,X). Although this was an error in the context of this puzzle, the error itself makes for an interesting program. It is a technique worth using in other programs. Discoveries like this should not be discarded as errors but saved for other programs where they might be useful.

## 5

```
10 INPUT"HOW OLD ARE YOU";X
20 PRINT"IN FIVE YEARS YOU WILL BE"
30 PRINTX+5

READY.
```

Notice that in the error program you INPUT A and then added +5. However, there was no X to add anything with. Your variables did not agree so the computer printed 5 no matter what age was entered. When line 10 was changed to:

```
10 INPUT "HOW OLD ARE YOU";X
```

the program worked. It is essential to check all of your variables and see that they are properly referenced to each other. This is a basic principle of programming.

## 6

### A

```
10 FORA=1TO2
20 FORB=1TO3
30 PRINT"HIP!"
40 NEXTB
50 PRINT"HOORAY!"
60 NEXTA

READY.
```

### B

```
10 FORA=1TO3
20 FORB=1TO2
30 PRINT"HOORAY!"
40 NEXTB
50 PRINT"HIP!"
60 NEXTA

READY.
```

**C**

```
10 FORA=1TO3
20 FORB=1TO2
30 PRINT"HIP!"
40 NEXTA
50 PRINT"HOORAY!"
60 NEXTB

READY.
```

Notice that the differences in these programs all have to do with the way in which FOR/NEXT loops are used. You always have to be careful that you go through the loop the number of times you plan to and that each FOR statement has a NEXT statement to continue and eventually close the loop. A simple FOR/NEXT mistake can turn an otherwise elegant program into a mess.


# Single Line Scrambles

The puzzles in this section consist of short programs with one line all scrambled up. There is a screen dump accompanying each program so that you can tell what the program is supposed to do. An ERROR statement marks the scrambled line so you don't have to figure it out for yourself. Before we begin, here are examples of scrambled and unscrambled lines:

Scrambled:

```
50 REM ERROR X=LET()()*RND10INT
```

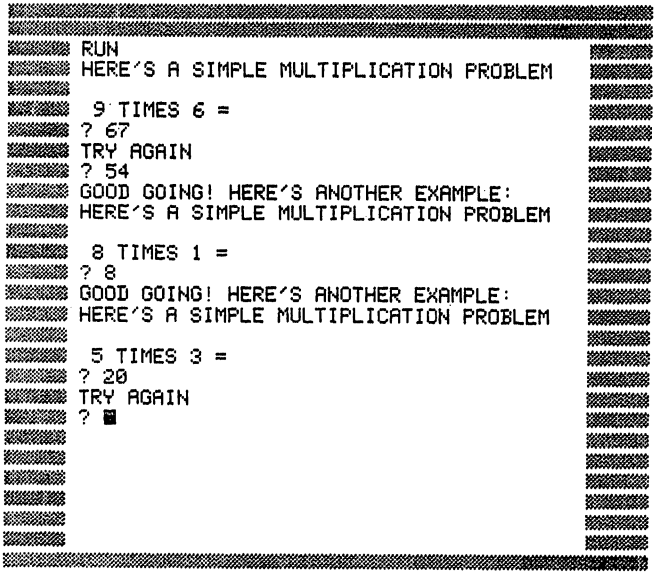
Unscrambled:

```
50 LET X=INT(RND(1)*10)
```

# 1

# SIMPLE MULTIPLICATION

We'll start with a simple multiplication program. As you can see from the screen dump, the program gives you multiplication problems as well as an opportunity to try again if you get the wrong answer.



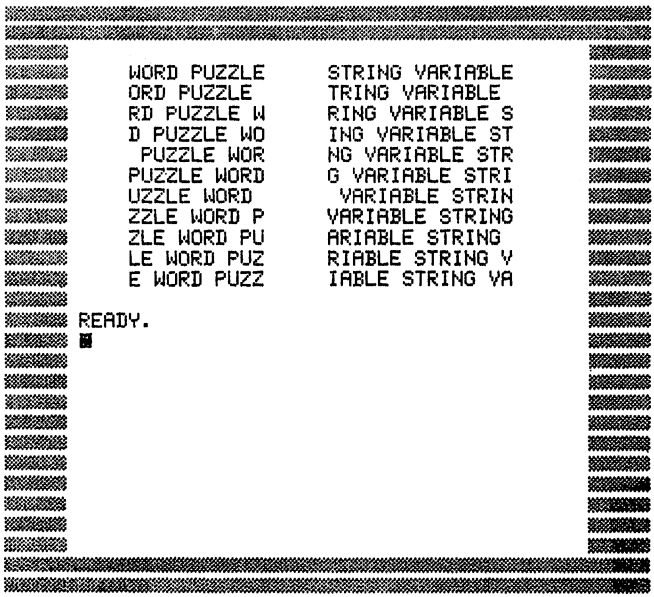

Here's the scrambled program:

```
10 PRINT "HERE'S A SIMPLE MULTIPLICATION PROBLEM"
20 PRINT
30 LETX=INT(RND(1)*10)
40 LETY=INT(RND(1)*10)
50 PRINTX;"TIMES";Y;"="
60 INPUTZ
70 REM ERROR- XYZ*THEN100IFGOTO=
80 PRINT"TRY AGAIN":GOTO60
100 PRINT"GOOD GOING! HERE'S ANOTHER EXAMPLE:"
110 GOTO10

READY.
```

# 2

# SPLIT SCREEN

This split screen word puzzle prints one thing on one side of the screen and another on the other side as illustrated in the screen dump:



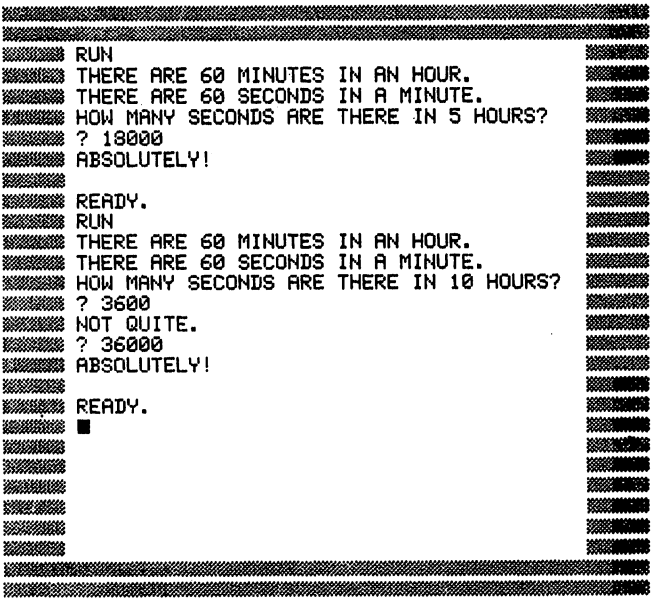

Here is the scrambled program:

```
10 LETA$=" WORD PUZZLE"
20 LETB$=" STRING VARIABLE"
30 PRINTCHR$(147)
40 FORX=11TO1STEP-1
50 PRINTTAB(4)RIGHT$(A$,X);LEFT$(A$,11-X);
60 REM ERROR- -+;LEFT$B$(((0)))TAB4PRINT11,X,XRIGHT$20B$
70 NEXT

READY.
```

# 3

# HOURS TO SECONDS

This simple program converts hours to seconds. It asks you how many seconds there are in any number of hours. It won't ask for seconds in more than 10 hours since the calculating gets boring at that point; nevertheless, it could.



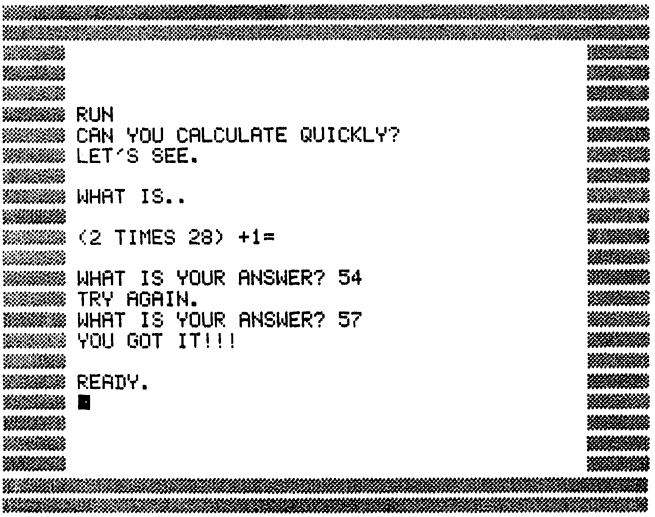

Here's the scrambled program:

```
10 PRINT"THERE ARE 60 MINUTES IN AN HOUR."
20 PRINT"THERE ARE 60 SECONDS IN A MINUTE."
30 LETX=INT(RND(1)*10)+1
40 PRINT"HOW MANY SECONDS ARE THERE IN"X"HOURS?"
50 INPUTY
60 REM ERROR- PRINT=THEN!END"60ABSOLUTELY60*IF*X":Y
70 PRINT"NOT QUITE.":GOTO50

READY.
```

# 4

# QUICK CALCULATION

This program sees if you can do simple number manipulations under some time pressure. It mixes multiplication with simple addition by asking you to add 1 to your times answer. Of course, any number could be added and the challenge could be made more complex.




Here is the scrambled program:

```
10 PRINT"CAN YOU CALCULATE QUICKLY?"
20 PRINT"LET'S SEE."
30 PRINT
40 PRINT"WHAT IS.."
50 LETX=INT(RND(1)*50)
60 PRINT
70 REM ERROR- ";;"+"="PRINT2XTIMES)CHR$(157)  (1
80 FORZ=1TO1500:NEXT
90 PRINT
100 INPUT"WHAT IS YOUR ANSWER";Y
110 IFY=(2*X)+1 THENPRINT "YOU GOT IT!!!":END
130 PRINT"TRY AGAIN.":GOTO100

READY.
```

# 5

# SIMPLE NUMBER COMPARISON

This is a simple number comparison game for young children. It asks whether one number is greater than another. If you look closely at the program you'll see that the computer will never select two equal numbers. It teaches the use of > and < .

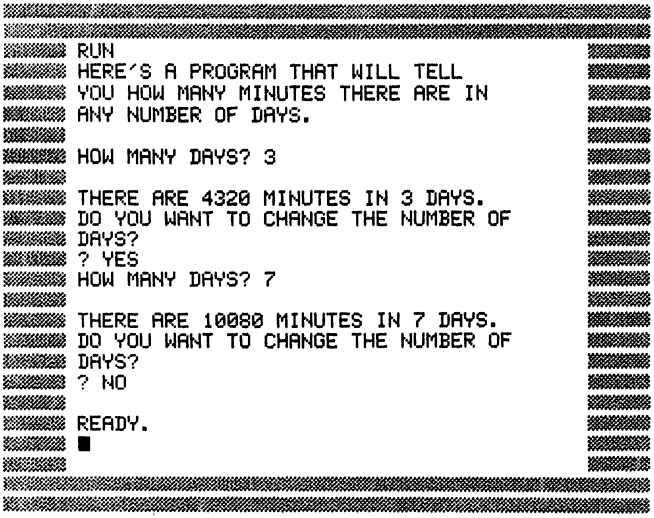
```
RUN
HERE'S A SIMPLE NUMBER GAME:
IS 10 GREATER OR LESS THAN 0

TYPE > IF IT IS GREATER AND < IF IT
IS LESS
? >
YOU GOT IT

READY.
RUN
HERE'S A SIMPLE NUMBER GAME:
IS 10 GREATER OR LESS THAN 8

TYPE > IF IT IS GREATER AND < IF IT
IS LESS
? ■
```

```
10 PRINT"HERE'S A SIMPLE NUMBER GAME:"
20 REM ERROR- X=RND(INT(20)+LET)*1
30 IFX=10THENGOTO20
40 PRINT"IS 10 GREATER OR LESS THAN";X
50 PRINT
60 PRINT"TYPE > IF IT IS GREATER AND < IF IT IS LESS"
70 INPUTA$
80 IFX<10THEN1000
90 IFX>10THEN2000
100 NEXT
1000 IFA$=">"THENPRINT"YOU GOT IT":END
1010 PRINT"SORRY.  TYPE RUN AND TRY AGAIN IF YOU LIKE.
     ":END
2000 IFA$="<"THENPRINT"YOU GOT IT":END
2010 PRINT"SORRY.  TYPE RUN AND TRY AGAIN IF YOU LIKE.
     ":END

READY.
```

# 6

# MINUTE/DAY CONVERSION

Here's a minute/day conversion program that tells you how many minutes there are in any number of days. It does not quiz you but answers your question instead.



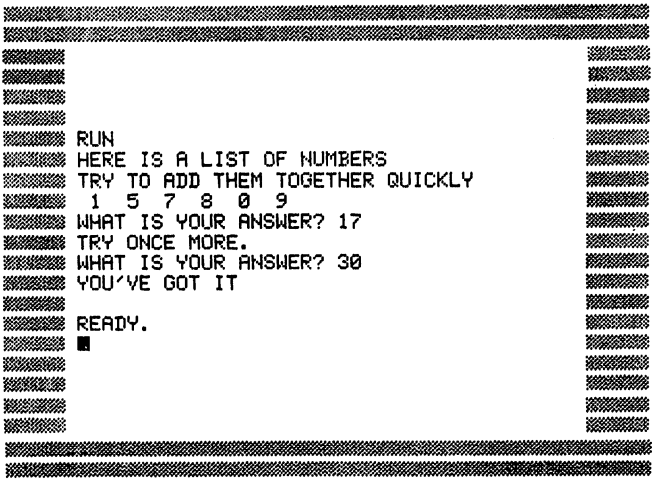

Here is the scrambled program:

```
10 PRINT"HERE'S A PROGRAM THAT WILL TELL"
20 PRINT"YOU HOW MANY MINUTES THERE ARE IN"
30 PRINT"ANY NUMBER OF DAYS."
40 PRINT
50 INPUT"HOW MANY DAYS";X
60 GOSUB500
70 PRINT"DO YOU WANT TO CHANGE THE NUMBER OF     DAYS?"
80 INPUTA$
90 IF A$="YES"THEN50
100 END
500 PRINT
510 REM ERROR- X"X60PRINT"THERE*MINUTES*DAYS""IN"ARE";24
520 RETURN

READY.
```

# 7

# NUMBER STRING

This puzzle challenges you to add a string of six numbers together quickly.



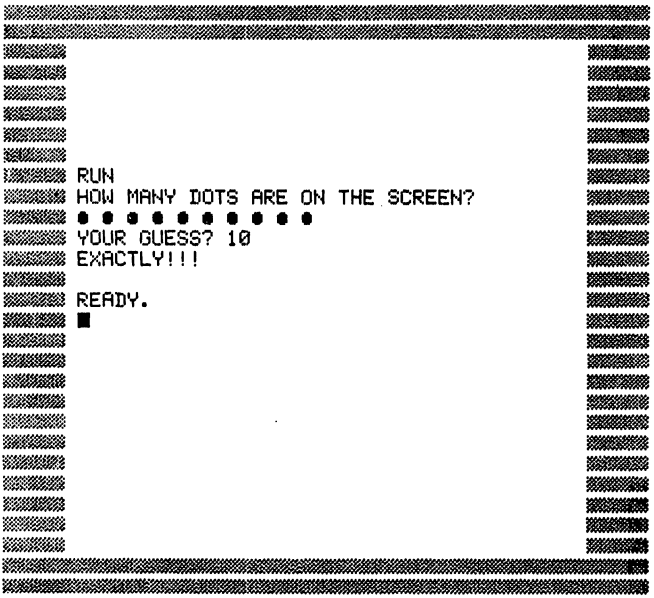

Here is the scrambled program:

```
10 PRINT"HERE IS A LIST OF NUMBERS"
20 PRINT"TRY TO ADD THEM TOGETHER QUICKLY"
30 FORZ=1TO1000:NEXT
40 DIMA(6)
50 FORX=1TO6
60 REM ERROR- AX(((RNDINT)1))10*=
70 PRINTA(X);
80 NEXT
90 FORZ=1TO2000:NEXT
100 PRINT
110 INPUT"WHAT IS YOUR ANSWER";Y
120 IFY=A(1)+A(2)+A(3)+A(4)+A(5)+A(6)THENPRINT"YOU'VE
    GOT IT":END
130 IFY<>A(1)+A(2)+A(3)+A(4)+A(5)+A(6)THENPRINT"TRY
    ONCE MORE.":GOTO110

READY.
```

# 8

# COUNT THE DOTS

This is a dot counting game.



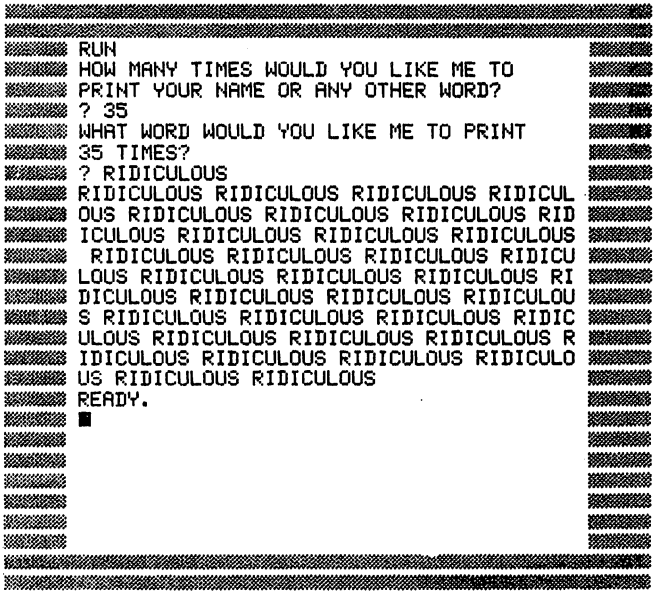

Here is the scrambled program:

```
10 PRINT "HOW MANY DOTS ARE ON THE SCREEN?"
20 LET X=INT(RND(1)*15)+1
30 FORY=1TOX
40 REM ERROR- " PRINT●";
50 NEXT
60 PRINT
70 INPUT"YOUR GUESS";Z
80 IFZ=XTHENPRINT"EXACTLY!!!":END
90 PRINT"COUNT AGAIN....":GOTO70

READY.
```

# 9

# RIDICULOUS

This program can print the word "RIDICULOUS" any number of times, perhaps a ridiculous venture but one whose structure has use in some programming contexts.




Here is the scrambled program:

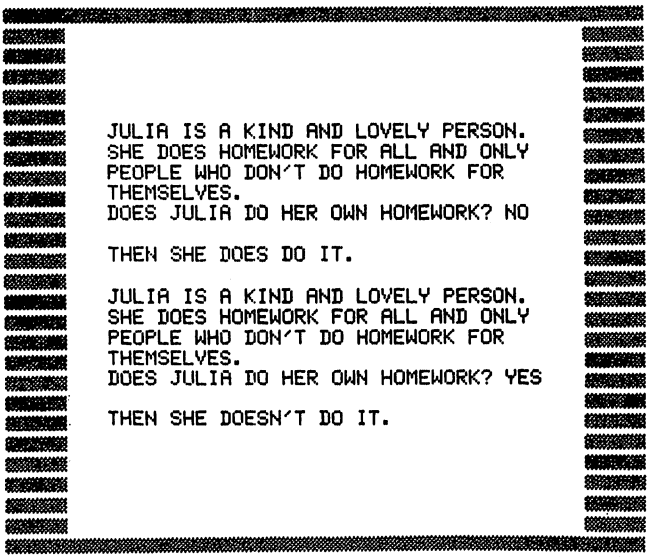
```
10 PRINT"HOW MANY TIMES WOULD YOU LIKE ME TO"
20 PRINT"PRINT YOUR NAME OR ANY OTHER WORD?"
30 INPUTX
40 PRINT"WHAT WORD WOULD YOU LIKE ME TO PRINT "X"TIMES?"
50 INPUT A$
60 REM ERROR- =XTO1YFOR
70 PRINTA$" ";
80 NEXT

READY.
```

# 10

# BARBER'S PARADOX

This little scramble is a version of the classical Barber's Paradox: There is a barber in a town who shaves all and only people in the town who don't shave themselves. Does he shave himself?

I've taken the paradox to school you might say:




Here is the scrambled program:

```
10 PRINT
20 PRINT"JULIA IS A KIND AND LOVELY PERSON."
30 PRINT"SHE DOES HOMEWORK FOR ALL AND ONLY"
40 PRINT"PEOPLE WHO DON'T DO HOMEWORK FOR"
50 PRINT"THEMSELVES.
60 INPUT"DOES JULIA DO HER OWN HOMEWORK";A$
70 PRINT
80 REM ERROR- 10 " .THENGOTO" A$ =DOESN'TIF"SHEYES
   :PRINT"ITDOTHEN
90 IFA$="NO"THENPRINT"THEN SHE DOES DO IT.":GOTO10

READY.
```

# Answers

## 1

```
10 PRINT "HERE'S A SIMPLE MULTIPLICATION PROBLEM"
20 PRINT
30 LETX=INT(RND(1)*10)
40 LETY=INT(RND(1)*10)
50 PRINTX;"TIMES";Y;"="
60 INPUTZ
70 IFZ=X*YTHENGOTO100
80 PRINT"TRY AGAIN":GOTO60
100 PRINT"GOOD GOING! HERE'S ANOTHER EXAMPLE:"
110 GOTO10

READY.
```

As you can see, line 70 (the scrambled line) checks for correct answers using the variable Z which is the answer you input. If Z=X*Y then you got the program and are sent to line 100 for congratulations and a chance at another example. If Z >< X*Y then the program moves to line 80 which sends you back to the problem you missed. This little technique can be incorporated in games where you want to give people many chances to answer a question as well as to generate new questions.

## 2

```
10 LETA$=" WORD PUZZLE"
20 LETB$=" STRING VARIABLE"
30 PRINTCHR$(147)
40 FORX=11TO1STEP-1
50 PRINTTAB(4)RIGHT$(A$,X);LEFT$(A$,11-X);
60 PRINTTAB(20)RIGHT$(B$,X+4);LEFT$(B$,11-X)
70 NEXT

READY.
```

The scrambled line 60 uses the TAB, RIGHT$, and LEFT$ commands. One strategy for going about unscrambling the line is to figure out how these commands are used in Commodore BASIC. For example, RIGHT$ and LEFT$ both take three variables separated by commas and surrounded by parentheses. This gives you a way of beginning to decipher the scrambling. Understanding structure is usually an aid to deciphering whether it has to do with line scrambles or secret codes.

## 3

```
10 PRINT"THERE ARE 60 MINUTES IN AN HOUR."
20 PRINT"THERE ARE 60 SECONDS IN A MINUTE."
30 LETX=INT(RND(1)*10)+1
40 PRINT"HOW MANY SECONDS ARE THERE IN"X"HOURS?"
50 INPUTY
60 IFY=X*60*60THENPRINT"ABSOLUTELY!":END
70 PRINT"NOT QUITE.":GOTO50

READY.
```

The key to unscrambling this puzzle is to know that you calculate the number of seconds in an hour using the formula X(the number of hours)*60*60 which is the conversion formula. This form of program can be changed to ask about converting feet or miles to inches, meters to centimeters, etc. It is a simple and generalizable form of a conversion quiz program.

## 4

```
10 PRINT"CAN YOU CALCULATE QUICKLY?"
20 PRINT"LET'S SEE."
30 PRINT
40 PRINT"WHAT IS.."
50 LETX=INT(RND(1)*50)
60 PRINT
```

```
70 PRINT"(2 TIMES";X;CHR$(157)") +1="
80 FORZ=1TO1500:NEXT
90 PRINT
100 INPUT"WHAT IS YOUR ANSWER";Y
110 IFY=(2*X)+1 THENPRINT "YOU GOT IT!!!":END
130 PRINT"TRY AGAIN.":GOTO100

READY.
```

The key to unscrambling this is to figure out that there is only one variable in this challenge. (2*X)+1 is the major part of the reconstruction of the scrambled line.

## 5

```
10 PRINT"HERE'S A SIMPLE NUMBER GAME:"
20 LETX=INT(RND(1)*20)
30 IFX=10THENGOTO20
40 PRINT"IS 10 GREATER OR LESS THAN";X
50 PRINT
60 PRINT"TYPE > IF IT IS GREATER AND < IF IT IS LESS"
70 INPUTA$
80 IFX<10THEN1000
90 IFX>10THEN2000
100 NEXT
1000 IFA$=">"THENPRINT"YOU GOT IT":END
1010 PRINT"SORRY.  TYPE RUN AND TRY AGAIN IF YOU LIKE.
     ":END
2000 IFA$="<"THENPRINT"YOU GOT IT":END
2010 PRINT"SORRY.  TYPE RUN AND TRY AGAIN IF YOU LIKE.
     ":END

READY.
```

This is simply an unscrambling of the integer random number function in Commodore BASIC. Some beginning programmers confuse RND(1)*X which gives a decimal answer with INT(RND(1)*X) which gives an integer. There are times when having integers is essential to a game or program. Also, it is important to pay attention to the placement of parentheses in the INT statement. A simple mistake like:

```
INT(RND(1))*X
```

can cause quite a mess. Try to figure out the different effects of INT(RND(1))*6). What you will see is the result of the fact that INT(RND(1)) is always 0.

## 6

```
10 PRINT"HERE'S A PROGRAM THAT WILL TELL"
20 PRINT"YOU HOW MANY MINUTES THERE ARE IN"
30 PRINT"ANY NUMBER OF DAYS."
40 PRINT
50 INPUT"HOW MANY DAYS";X
60 GOSUB500
70 PRINT"DO YOU WANT TO CHANGE THE NUMBER OF     DAYS?"
80 INPUTA$
90 IF A$="YES"THEN50
100 END
500 PRINT
510 PRINT"THERE ARE"X*60*24"MINUTES IN"X"DAYS."
520 RETURN

READY.
```

Line 510 combines calculation with printing and, like the previous conversion puzzle (number 3), can be used to perform many different types of conversions. In unscrambling this the thing to watch for is the placement of quotes and semicolons. A misplaced quote or semicolon can destroy an entire program.

## 7

```
10 PRINT"HERE IS A LIST OF NUMBERS"
20 PRINT"TRY TO ADD THEM TOGETHER QUICKLY"
30 FORZ=1TO1000:NEXT
40 DIMA(6)
```

```
50 FORX=1TO6
60 A(X)=INT(RND(1)*10)
70 PRINTA(X);
80 NEXT
90 FORZ=1TO2000:NEXT
100 PRINT
110 INPUT"WHAT IS YOUR ANSWER";Y
120 IFY=A(1)+A(2)+A(3)+A(4)+A(5)+A(6)THENPRINT"YOU'VE
    GOT IT":END
130 IFY<>A(1)+A(2)+A(3)+A(4)+A(5)+A(6)THENPRINT"TRY
    ONCE MORE.":GOTO110

READY.
```

This program makes use of a powerful aspect of Commodore BASIC, the array. A(X) is dimensioned on line 40 to contain six numbers. The scrambled line 60 places a random integer between 0 and 10 in each position. That way, lists of numbers can be generated and used as in this program. Beginning and intermediate programmers should take advantage of the ability to store and use lists that this function provides. For examples of the use of arrays see your **Commodore BASIC Reference Manual** or look up arrays in a text if you are not already familiar with them.

## 8

```
10 PRINT "HOW MANY DOTS ARE ON THE SCREEN?"
20 LET X=INT(RND(1)*15)+1
30 FORY=1TOX
40 PRINT"● ";
50 NEXT
60 PRINT
70 INPUT"YOUR GUESS";Z
80 IFZ=XTHENPRINT"EXACTLY!!!":END
90 PRINT"COUNT AGAIN....":GOTO70

READY.
```

This program is an example of how you can use SHIFT graphics characters (in the case of this program **SHIFT Q**) in your program. It is worth exploring this extra graphics keyboard that your Commodore provides for you. It is the easiest of the different graphics modes that are available to you and can create interesting graphics with some experimentation.

## 9

```
10 PRINT"HOW MANY TIMES WOULD YOU LIKE ME TO"
20 PRINT"PRINT YOUR NAME OR ANY OTHER WORD?"
30 INPUTX
40 PRINT"WHAT WORD WOULD YOU LIKE ME TO PRINT "X"TIMES?"
50 INPUT A$
60 FORY=1TOX
70 PRINTA$" ";
80 NEXT

READY.
```

This is a scramble of the FOR statement of the FOR/NEXT loop used in the program to produce the ridiculous repetition. Notice that the number of times through the loop (the X variable) changes each time you run the program. Most beginning programmers use fixed loops like FOR X=1 TO 59; but, sometimes it is more useful to change the number of times through the loop as the program is running.

## 10

```
10 PRINT
20 PRINT"JULIA IS A KIND AND LOVELY PERSON."
30 PRINT"SHE DOES HOMEWORK FOR ALL AND ONLY"
40 PRINT"PEOPLE WHO DON'T DO HOMEWORK FOR"
50 PRINT"THEMSELVES.
60 INPUT"DOES JULIA DO HER OWN HOMEWORK";A$
70 PRINT
80 IFA$="YES"THENPRINT"THEN SHE DOESN'T DO IT.":GOTO10
90 IFA$="NO"THENPRINT"THEN SHE DOES DO IT.":GOTO10

READY.
```

The scrambled line 80, as well as line 90, make the paradox continue until you are bored and shut down the computer. However, these lines illustrate a technique you can add to many of the games and quizzes in this section. It will allow them to run many times instead of ending after just one play.

Here's a very simple example:

```
10 PRINT "WHAT NUMBER IS THIS?"
20 LET X=INT(RND(1)*20)
30 INPUT Y
40 IF X=Y THEN PRINT"GREAT":GOTO 10
50 IF X <>Y THEN PRINT"TRY AGAIN":GOTO 30
```

This way you have many guesses and can take a new turn once you guess correctly.


# Line Number Scrambles

The puzzles in this section leave all the codes in a program intact. All they do is scramble the line numbers. These numbers are not scrambled according to any pattern, so there is no use trying to solve the puzzles by trying to figure out a pattern for transforming the line numbers. You have to pay attention to the screen dumps and try to put a program structure together that will do what the illustrations show.

Scrambling line numbers of even the simplest programs can destroy them. For example, take this two-line program:

```
10 PRINT "HO HO HO!";
20 GOTO 10
```

This will fill the screen up with a jolly greeting. Reverse the line numbers and you get:

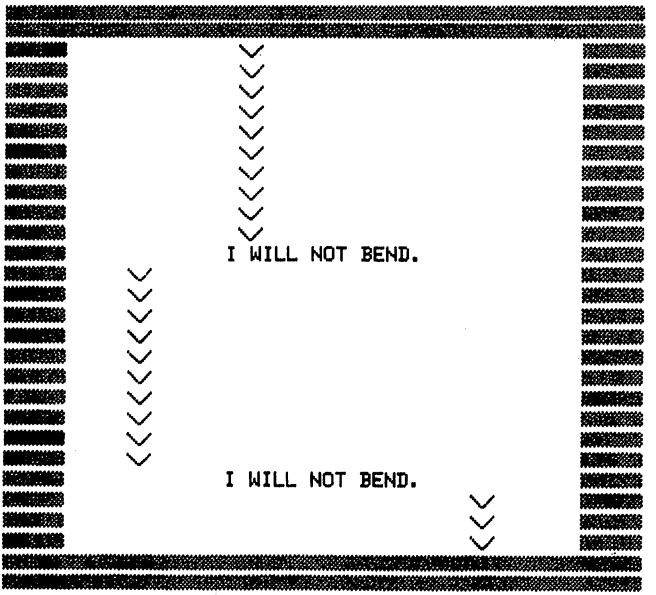
```
10 GOTO 10
20 PRINT "HO HO HO!";
```

This program will lock up at line 10 and stay there until you press the **BREAK** key. Nothing will happen. In the simplest way, this is an illustration of the importance of program structure in BASIC. Now for the puzzles.

# 1

# I WILL NOT BEND

This program prints a V shape and moves relentlessly down the screen printing "I will not bend," every 10 repeats of the V-form. Here is the screen dump and a scrambled version of the program. Unscramble the program and, if you feel like it, make it fancier.

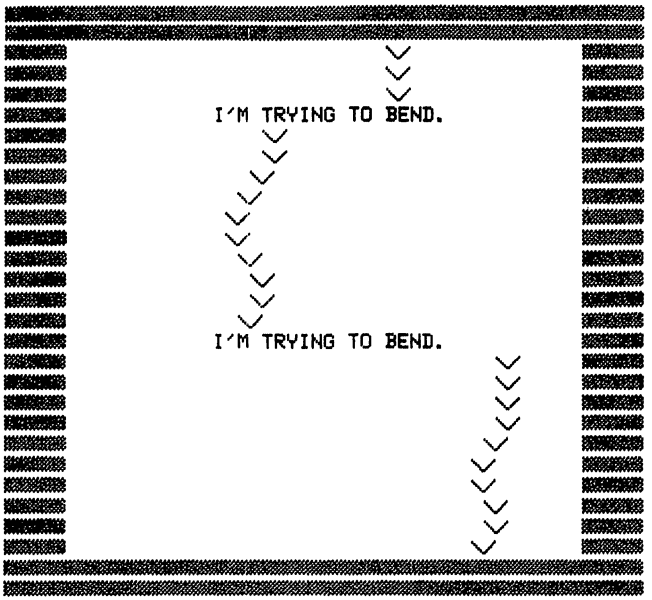



```
10 FORY=1TO10
20 NEXT
30 PRINTTAB(X)"\/"
40 GOTO10
50 X=INT(RND(1)*38)
60 PRINTTAB(12)"I WILL NOT BEND."

READY.
```

# 2

# I'M TRYING TO BEND

This simple variant on puzzle 1 produces a completely different result. You should be able to think through the difference and reconstruct the program that makes a bit of a curve.

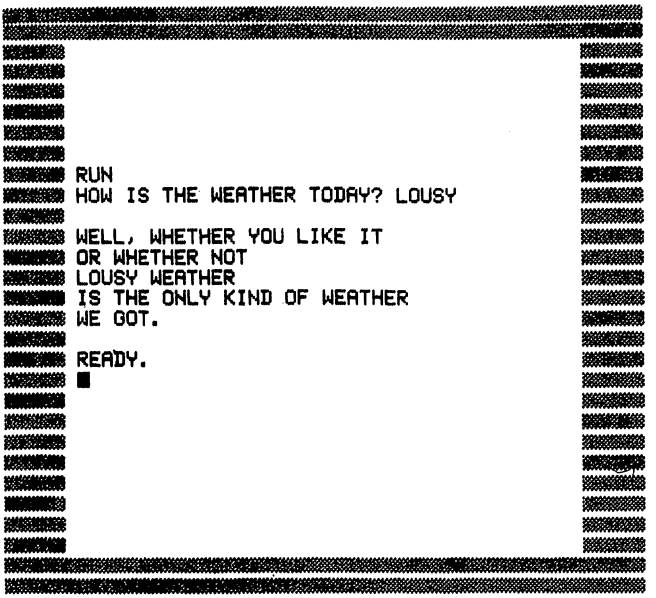



```
10 X=X+INT(RND(1)*3)-1
20 IFX<0THENX=38
30 FORY=1TO10
40 PRINTTAB(11)"I'M TRYING TO BEND."
50 GOTO10
60 X=INT(RND(1)*38)
70 NEXT
80 IFX>38THENX=0
90 PRINTTAB(X)"\/"

READY.
```

# 3

# A POETIC SCRAMBLE

Here is a little poem with some input about the weather. The input combined with the poem should help you reassemble the scrambled program.

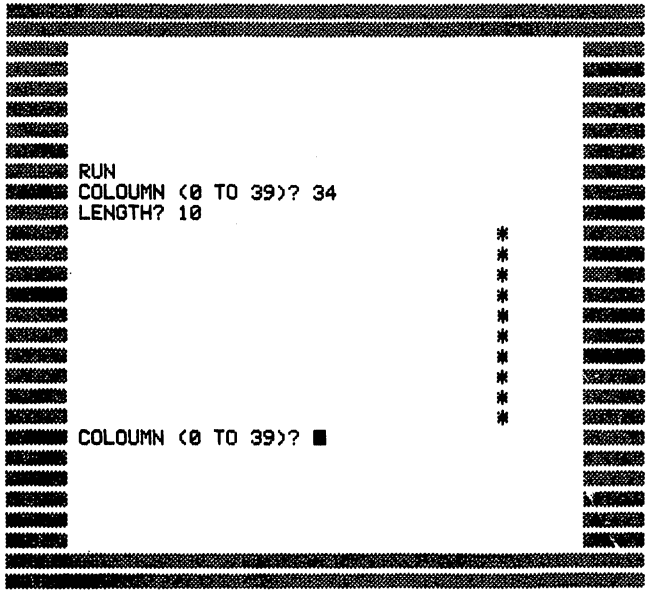



```
10 PRINTA$" WEATHER"
20 PRINT"WELL, WHETHER YOU LIKE IT"
30 PRINT"IS THE ONLY KIND OF WEATHER"
40 PRINT"OR WHETHER NOT"
50 PRINT"WE GOT."
60 PRINT
70 INPUT"HOW IS THE WEATHER TODAY";A$

READY.
```

# 4

# WHERE AM I?

This program allows you to pick a column on the screen from 1 to 39 and then pick the number of stars you want to print out. The result of picking column 34 with length of 10 is this:

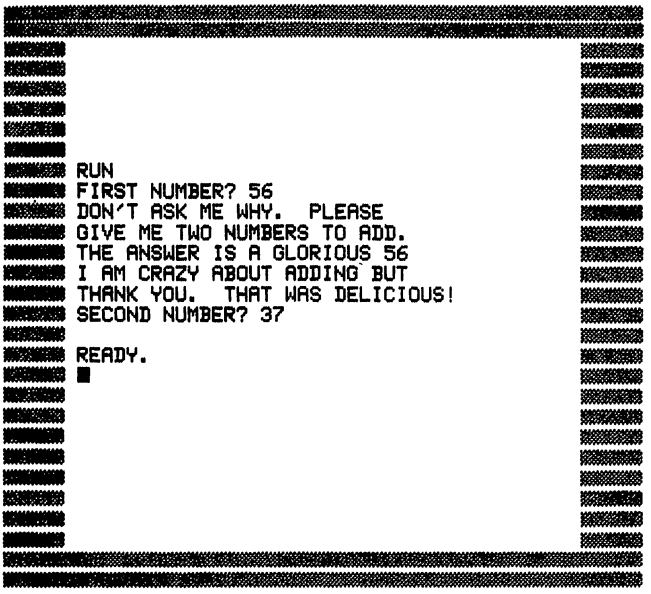



```
10 NEXT
20 GOTO10
30 INPUT"LENGTH";B
40 FORC=1TOB
50 INPUT"COLOUMN (0 TO 39)";A
60 PRINTTAB(A)"*"

READY.
```

# 5

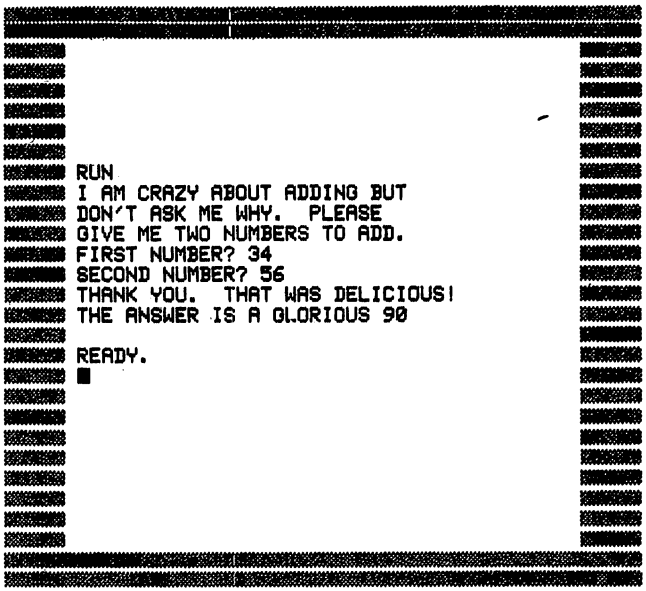
# A LITTLE DIFFERENT ORDER

Look at this scrambled screen dump:




Here is the scrambled program and a dump of what the unscrambled program is supposed to do. What is the unscrambled program?

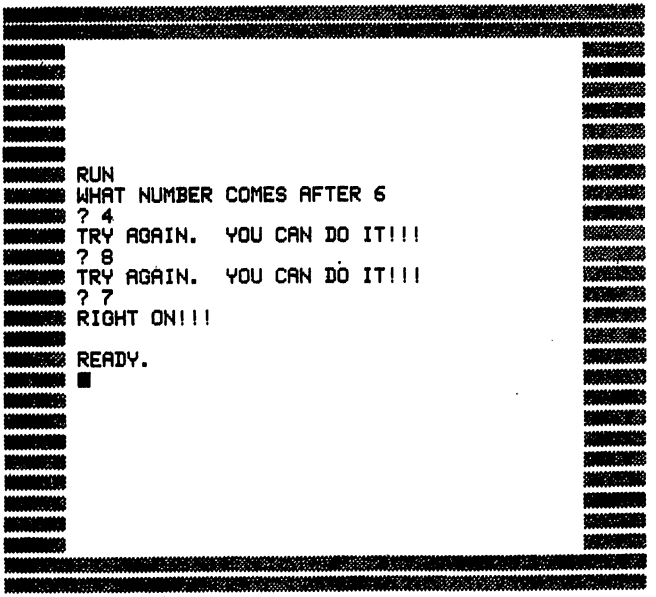
```
10 INPUT"FIRST NUMBER";X
20 PRINT"DON'T ASK ME WHY.  PLEASE"
30 PRINT"GIVE ME TWO NUMBERS TO ADD."
40 PRINT"THE ANSWER IS A GLORIOUS";X+Y
50 PRINT"I AM CRAZY ABOUT ADDING BUT"
60 PRINT"THANK YOU.  THAT WAS DELICIOUS!"
70 INPUT"SECOND NUMBER";Y

READY.
```


RUN
I AM CRAZY ABOUT ADDING BUT
DON'T ASK ME WHY. PLEASE
GIVE ME TWO NUMBERS TO ADD.
FIRST NUMBER? 34
SECOND NUMBER? 56
THANK YOU. THAT WAS DELICIOUS!
THE ANSWER IS A GLORIOUS 90
READY.

# 6

# CAN YOU COUNT?

This program picks a number and asks you what comes next.

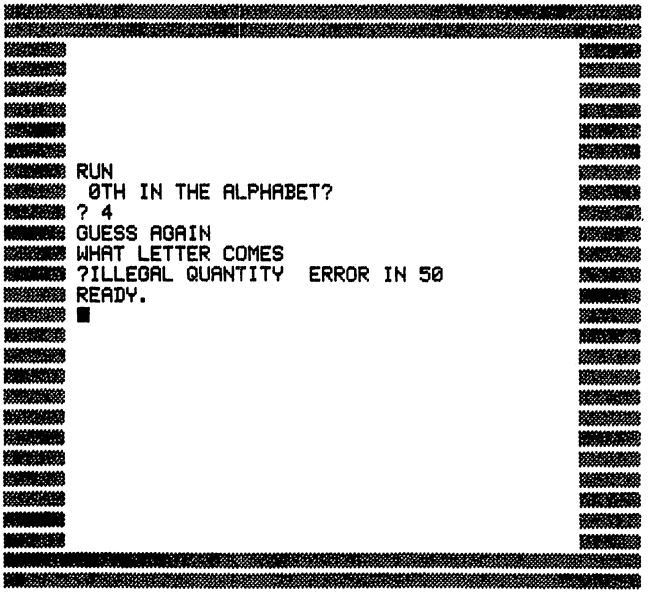



```
10 PRINT"WHAT NUMBER COMES AFTER";X
20 IFY<>X+1THENPRINT"TRY AGAIN.  YOU CAN DO IT!!!"
30 LETX=INT(RND(1)*11)
40 IFY=X+1THENPRINT"RIGHT ON!!!":END
50 GOTO30
60 INPUTY

READY.
```

# 7

# NEXT LETTER?

This program is supposed to ask what letter comes next in the alphabet after one chosen by the computer. When scrambled here is what it does:




Here is the scrambled program and a screen dump of it running unscrambled:

```
10 PRINTXCHR$(157)"TH IN THE ALPHABET?"
20 INPUTB$
30 PRINT"GUESS AGAIN"
40 PRINT"WHAT LETTER COMES";
50 IFMID$(A$,X,1)=B$THENPRINT"RIGHT":END
60 A$="ABCDEFGHIJKLMNOPQRSTUVWXYZ"
70 GOTO50
80 LETX=INT(RND(1)*26)+1

READY.
```

```
RUN
WHAT LETTER COMES 15TH IN THE ALPHABET?
? N
GUESS AGAIN
? O
RIGHT

READY.
```

# 8

# NAME NAME

This program asks for your name and then prints it out 24 times. Here is what the program looks like on the screen as well as the scrambled version of the program and a dump of how this version runs. Unscramble the program and print out a friend's name.

```
RUN
WHAT IS YOUR NAME?
 WHAT A NICE NAME!
WHAT IS YOUR NAME?
 WHAT A NICE NAME!
WHAT IS YOUR NAME?
 WHAT A NICE NAME!
WHAT IS YOUR NAME?
 WHAT A NICE NAME!
WHAT IS YOUR NAME?
 WHAT A NICE NAME!
WHAT IS YOUR NAME?
 WHAT A NICE NAME!
WHAT IS YOUR NAME?
 W
```

```
WHAT IS YOUR NAME?
? JOSHUA
WHAT A NICE NAME!

JOSHUA JOSHUA JOSHUA JOSHUA JOSHUA JOSHU
A JOSHUA JOSHUA JOSHUA JOSHUA JOSHUA JOS
HUA JOSHUA JOSHUA JOSHUA JOSHUA JOSHUA J
OSHUA JOSHUA JOSHUA JOSHUA JOSHUA JOSHUA
 JOSHUA
READY.
```

```
10 FORZ=1TO24
20 PRINT"WHAT IS YOUR NAME?"
30 PRINTA$" ";
40 PRINT"WHAT A NICE NAME!"
50 NEXT
60 PRINTCHR$(147)
70 INPUTA$
80 PRINT

READY.
```

# 9

# MEDITATION

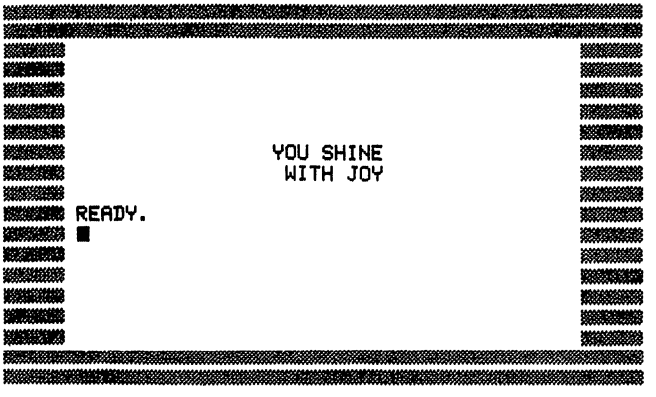
A screen dump can hardly capture this colorful program so you will have to imagine how this looks in Commodore color. Notice that this program uses some commands you may not be familiar with. BRD is the border color variable, BCK is the background color variable, 53280 is the border color memory location, and 53281 is the background color location. In order to make the program have a complex color cycle, the border, background, and type colors have all been used. Here is a screen dump of the program. Imagine that it is in color and unscramble the program:

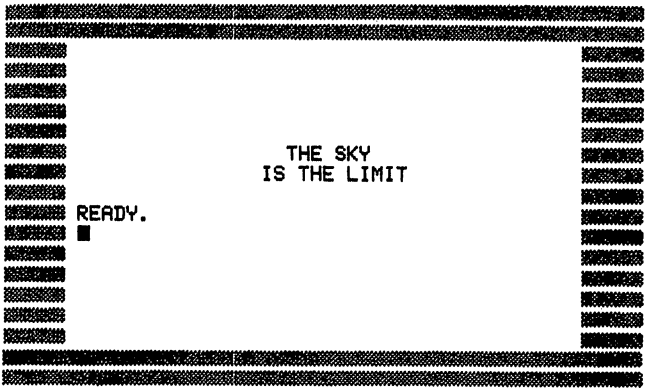

```
10 POKE53281,BCK
20 POKE53280,BRD
30 BCK=INT(RND(1)*15)
40 PRINT:PRINT:PRINT:PRINT:PRINT
50 FORX=1TO500:NEXT
60 PRINTTAB(7)"HERE ARE SOME COLORS FOR"
70 GOTO50
80 FORX=1TO500:NEXT
90 PRINTCHR$(147)CHR$(5)
100 BRD=INT(RND(1)*15)
110 PRINTTAB(7)"MEDITATION AND RELAXATION"

READY.
```

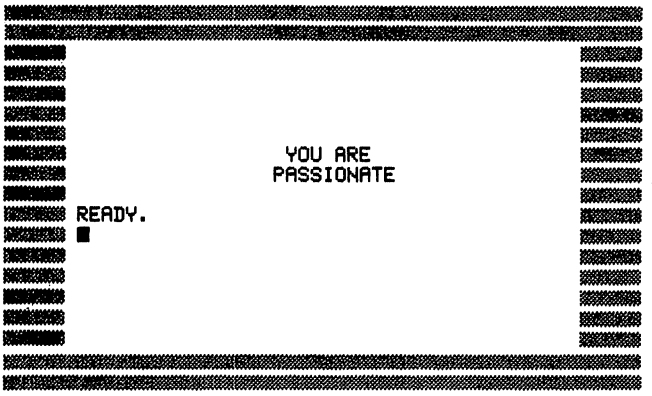
# 10

# AN OPTIMISTIC PSYCHIATRIST'S PROJECTIVE TEST

This test asks you for a color preference. All the responses are in full color and are positive because the psychiatrist that taught the test to me never found a person she didn't believe in. The black and white can't show you the passionate red, the sky blue, or the luminescent yellow but, once you get the program straightened out, you'll see them. Try to unscramble the program and then add color upon color until you can make positive statements corresponding to all the colors that can be produced by your Commodore 64.




```
10 PRINTTAB(17)"WITH JOY"
20 IFD$=B$THEN2000
30 PRINT:PRINT:PRINT:PRINT
40 PRINTTAB(17)"THE SKY"
50 POKE53281,2
60 POKE53281,0:PRINTCHR$(5)
70 PRINTTAB(15)"IS THE LIMIT"
80 END
90 PRINTTAB(17)"YOU ARE"
100 PRINT
110 PRINT:PRINT:PRINT:PRINT
1000 PRINT"        RED      BLUE      YELLOW"
1010 PRINTTAB(16)"YOU SHINE"
1020 A$="RED":B$="BLUE":C$="YELLOW"
1030 PRINT:PRINT:PRINT:PRINT
1040 END
```

```
2000 POKE53281,14
2010 PRINT"FAVORITE......"
2020 IFD$=A$THEN1000
2030 PRINT"WHICH OF THESE THREE COLORS IS YOUR"
2040 PRINTTAB(16)"PASSIONATE"
3000 POKE53281,7
3010 IFD$=C$THEN3000
3020 INPUTD$
3030 PRINTCHR$(147)
3040 END

READY.
```

# Answers

## 1

```
10 PRINTTAB(12)"I WILL NOT BEND."
20 X=INT(RND(1)*38)
30 FORY=1TO10
40 PRINTTAB(X)"\/"
50 NEXT
60 GOTO10

READY.
```

The key to unscrambling this is in figuring out where the GOTO statement must be placed. It gives you a sense of reference back and forward in the program and helps you reconstruct the original. In fact, try this same program with the following simple line change and see what happens:

```
60 GOTO 40
```

## 2

```
10 PRINTTAB(11)"I'M TRYING TO BEND."
20 X=INT(RND(1)*38)
30 FORY=1TO10
40 PRINTTAB(X)"\/"
50 X=X+INT(RND(1)*3)-1
60 IFX>38THENX=0
70 IFX<0THENX=38
80 NEXT
90 GOTO10

READY.
```

Compare the unscrambled version of this program with that of puzzle 1. Notice how line 50 causes the bending. In computer algebra, you can change the values of the variables in mid-program. A statement like LET X=X+1 is perfectly legitimate in computer algebra and is not valid for any X in the kind of algebra you learned in school.

## 3

```
10 INPUT"HOW IS THE WEATHER TODAY";A$
20 PRINT
30 PRINT"WELL, WHETHER YOU LIKE IT"
40 PRINT"OR WHETHER NOT"
50 PRINTA$" WEATHER"
60 PRINT"IS THE ONLY KIND OF WEATHER"
70 PRINT"WE GOT."

READY.
```

Almost any saying can be turned into a fun program using this form. Try an input to the following:

Too many X's spoil the Y.

A X in time saves Y.

Better X than Y.

## 4

```
10 INPUT"COLUMN (0 TO 39)";A
20 INPUT"LENGTH";B
30 FORC=1TOB
40 PRINTTAB(A)"*"
50 NEXT
60 GOTO10

READY.
```

After you understand how this program works, try to change the positioning of the stars. See if you can make them print out from left to right instead of top to bottom by having the player pick a row instead of a column.

## 5

```
10 PRINT"I AM CRAZY ABOUT ADDING BUT"
20 PRINT"DON'T ASK ME WHY.  PLEASE"
30 PRINT"GIVE ME TWO NUMBERS TO ADD."
40 INPUT"FIRST NUMBER";X
50 INPUT"SECOND NUMBER";Y
60 PRINT"THANK YOU.  THAT WAS DELICIOUS!"
70 PRINT"THE ANSWER IS A GLORIOUS";X+Y

READY.
```

Of course, this program can easily be extended to multiplication. See if you can get a bit more complex and extend it to division so the answer comes out even and to subtraction so the answer comes out positive. A few more lines of code will be required to guarantee these conditions.

## 6

```
10 LETX=INT(RND(1)*11)
20 PRINT"WHAT NUMBER COMES AFTER";X
30 INPUTY
40 IFY=X+1THENPRINT"RIGHT ON!!!":END
50 IFY<>X+1THENPRINT"TRY AGAIN.  YOU CAN DO IT!!!"
60 GOTO30

READY.
```

Here are some interesting modifications:

Can you tell me what is 5 times X?

Can you tell me what is 5 times X divided by three? (In this case set the program up so that the answer comes out even.)

## 7

```
10 A$="ABCDEFGHIJKLMNOPQRSTUVWXYZ"
20 LETX=INT(RND(1)*26)+1
30 PRINT"WHAT LETTER COMES";
40 PRINTXCHR$(157)"TH IN THE ALPHABET?"
50 INPUTB$
60 IFMID$(A$,X,1)=B$THENPRINT"RIGHT":END
70 PRINT"GUESS AGAIN"
80 GOTO50

READY.
```

Notice that the alphabet is stored in this program in A$ which was set equal to "ABCDEFGHIJKLMNOPQRSTUVWXYZ." Also notice that the key to the program is the powerful BASIC command MID$(A$,X,1) which allows you to choose the Xth member of A$. This is used in many code and word programs. You can use it to choose any number of letters out of a string. Here's an example of how it might be used:

```
Set A$="HIERARCHY"
Then MID$(A$,5,8)
```

will produce the word "arch." You can use this substring extraction to set up games that ask people to find out what words are contained within a given word.

## 8

```
10 PRINTCHR$(147)
20 PRINT"WHAT IS YOUR NAME?"
30 INPUTA$
40 PRINT"WHAT A NICE NAME!"
50 PRINT
60 FORZ=1TO24
70 PRINTA$" ";
80 NEXT

READY.
```

## 9

```
10 PRINTCHR$(147)CHR$(5)
20 PRINT:PRINT:PRINT:PRINT:PRINT
30 PRINTTAB(7)"HERE ARE SOME COLORS FOR"
40 PRINTTAB(7)"MEDITATION AND RELAXATION"
50 BRD=INT(RND(1)*15)
60 BCK=INT(RND(1)*15)
70 POKE53280,BRD
80 FORX=1TO500:NEXT
90 POKE53281,BCK
100 FORX=1TO500:NEXT
110 GOTO50

READY.
```

Play with the border and background colors. They can be used to dress up your programs and provide interesting detail to games and other things you program. One way to start is to cycle through all of the border and background colors and see what they do on the screen. Just play around with them and you will most likely find effects you can use in many different ways.

## 10

```
10 POKE53281,0:PRINTCHR$(5)
20 A$="RED":B$="BLUE":C$="YELLOW"
30 PRINT"WHICH OF THESE THREE COLORS IS YOUR"
40 PRINT"FAVORITE......"
50 PRINT
60 PRINT"        RED      BLUE      YELLOW"
70 INPUTD$
80 PRINTCHR$(147)
90 IFD$=A$THEN1000
100 IFD$=B$THEN2000
110 IFD$=C$THEN3000
1000 POKE53281,2
1010 PRINT:PRINT:PRINT:PRINT
1020 PRINTTAB(17)"YOU ARE"
1030 PRINTTAB(16)"PASSIONATE"
1040 END
2000 POKE53281,14
2010 PRINT:PRINT:PRINT:PRINT
```

```
2020 PRINTTAB(17)"THE SKY"
2030 PRINTTAB(15)"IS THE LIMIT"
2040 END
3000 POKE53281,7
3010 PRINT:PRINT:PRINT:PRINT
3020 PRINTTAB(16)"YOU SHINE"
3030 PRINTTAB(17)"WITH JOY"
3040 END

READY.
```

Here are a few color character equivalents I've managed to come up with:

orange—abundant and nourishing

pink—shy but perceptive

black—powerful and intelligent

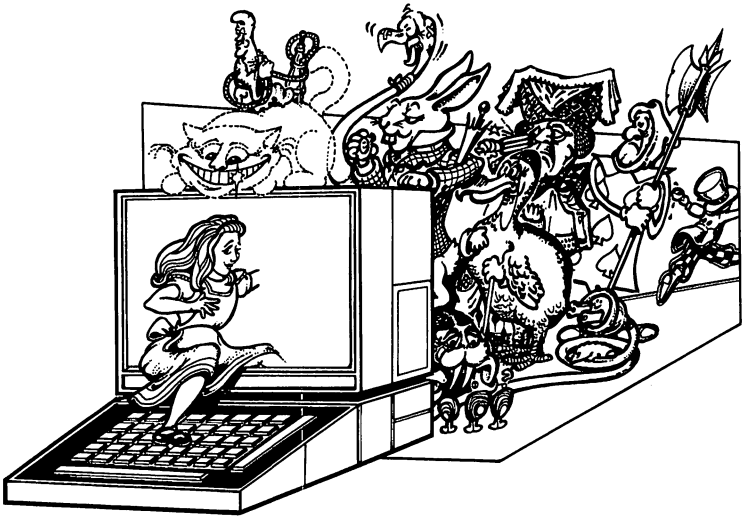
green—fruitful and abundant

dark blue—deep and curious

Surely you can come up with dozens more!

# Missing Line Puzzles

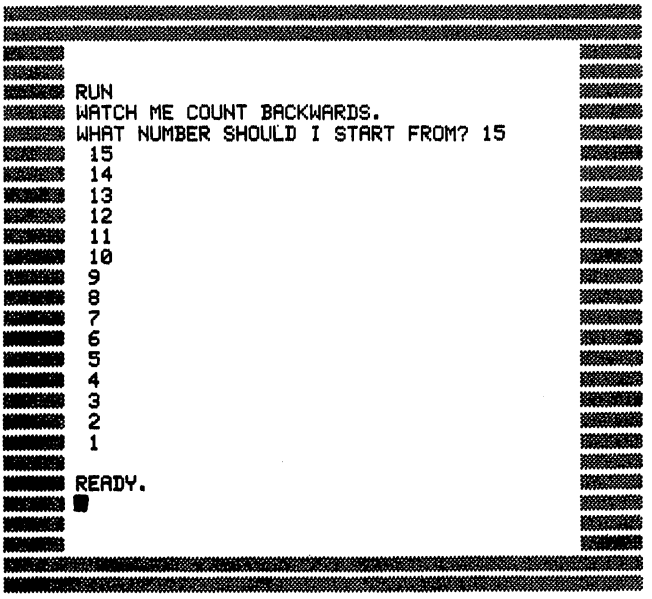
In these puzzles, one line is missing from the programs. The missing line is indicated by the line number and the statement PRINT "??????????????". In choosing what lines to drop out, the main consideration was to leave out some essential part of the program structure. In effect, these puzzles are exercises in reading and understanding program structure. The screen dumps should give you enough of an idea of how the program runs to allow you to think your way through to the full program. If you come up with a line that works and is not the one given in the answer please write and let me know. There are many ingenious ways to solve programming problems.



# 1

# COUNTING BACKWARDS

This program asks you for an integer and then counts backwards from that number to 1 and prints out the list.

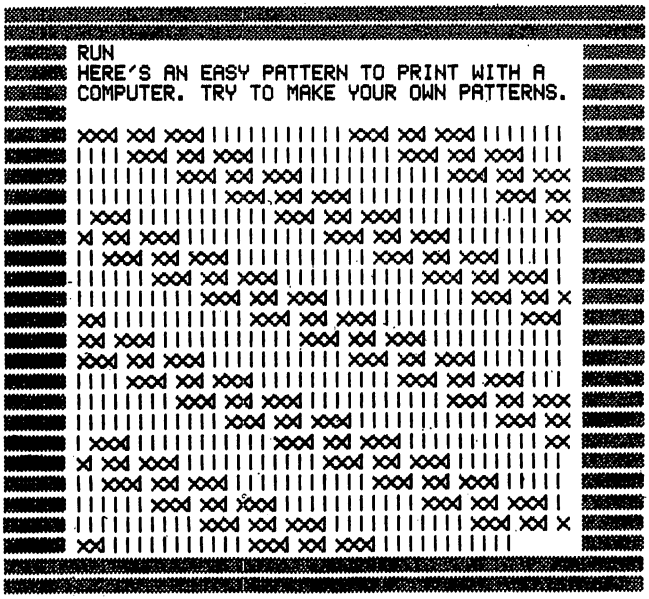



```
10 PRINT"WATCH ME COUNT BACKWARDS."
20 INPUT"WHAT NUMBER SHOULD I START FROM";X
30 PRINT"??????????????????????"
40 PRINTZ
50 NEXTZ

READY.
```

# 2

# PATTERN MAKING

This program prints out a pattern. One aspect of the pattern is left out for you to figure out.

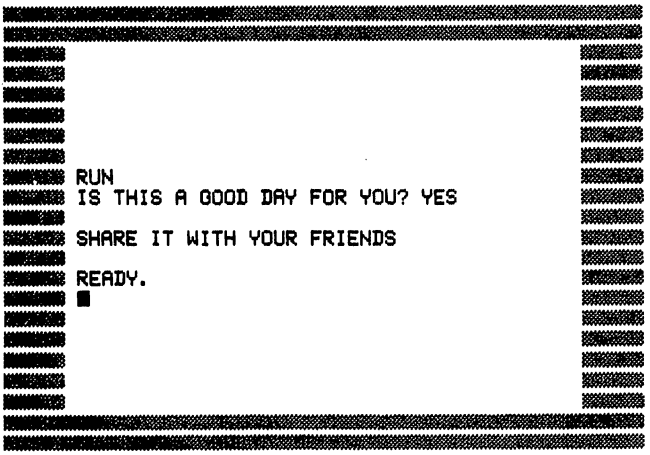



```
10 PRINT"HERE'S AN EASY PATTERN TO PRINT WITH A"
20 PRINT"COMPUTER. TRY TO MAKE YOUR OWN PATTERNS."
30 FORX=1TO750:NEXT
40 PRINT"??????????????????????"
50 PRINT"|||||||||||";
60 GOTO40

READY.
```

# 3

# A GOOD DAY

This program asks you if it's a good day for you. Here are the responses:




Here is the program with the missing line of code:

```
10 INPUT"IS THIS A GOOD DAY FOR YOU";A$
20 PRINT
30 IFA$="YES"THENPRINT"SHARE IT WITH YOUR FRIENDS":END
40 IF A$="NO"THENGOTO100
100 PRINT"MAYBE THIS WILL MAKE YOU FEEL BETTER.."
110 FORX=1TO1500:NEXT
120 PRINTTAB(RND(1)*20)"/";
130 PRINT"??????????????????????"
140 GOTO120

READY.
```

# 4

# GUESS

This is a letter counting game. It flashes a word on the screen and asks you to guess how many letters it has. You can count the letters but estimating the length of the word is more fun.

```
RUN
HERE ARE SOME WORDS TO STUDY FOR A FEW
SECONDS.  GUESS HOW  MANY  LETTERS  ARE
IN EACH WORD.

HOPEFUL
YOUR GUESS? 7
GREAT! TRY ANOTHER

INDEPENDENCE
YOUR GUESS? 12
GREAT! TRY ANOTHER

TROUBLE
YOUR GUESS? 8
TAKE ANOTHER LOOK

TROUBLE
YOUR GUESS? 7
GREAT! YOU GOT ALL THREE!

READY.
```

Here is the program with the missing line of code:

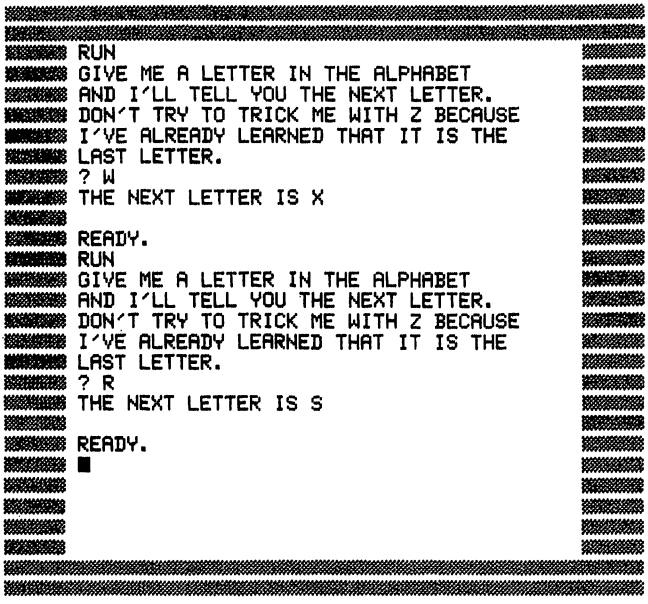
```
10 A$="HOPEFUL":B$="INDEPENDENCE":C$="TROUBLE"
20 PRINT"HERE ARE SOME WORDS TO STUDY FOR A FEW"
30 PRINT"SECONDS.  GUESS HOW  MANY  LETTERS  ARE"
40 PRINT"IN EACH WORD."
50 FORX=1TO750:NEXT
100 PRINT
110 PRINTA$
120 INPUT"YOUR GUESS";W
130 PRINT"??????????????????"
140 IFW<>LEN(A$)THENPRINT"TAKE ANOTHER LOOK":FORX=1TO
    750:NEXT:GOTO100
```

```
200 PRINT
210 PRINTB$
220 INPUT"YOUR GUESS";W
230 IFW=LEN(B$)THENPRINT"GREAT! TRY ANOTHER":FORX=1TO
    750:NEXT:GOTO300
240 IFW<>LEN(B$)THENPRINT"TAKE ANOTHER LOOK":FORX=1TO
    750:NEXT:GOTO200
300 PRINT
310 PRINTC$
320 INPUT"YOUR GUESS";W
330 IFW=LEN(C$)THENPRINT"GREAT! YOU GOT ALL THREE!":END
340 IFW<>LEN(C$)THENPRINT"TAKE ANOTHER LOOK":FORX=1TO
    750:NEXT:GOTO300

READY.
```

# 5

# NEXT LETTER

This program asks for a letter and then gives you the next letter in the alphabet. It won't be tricked if you ask for the letter after Z.



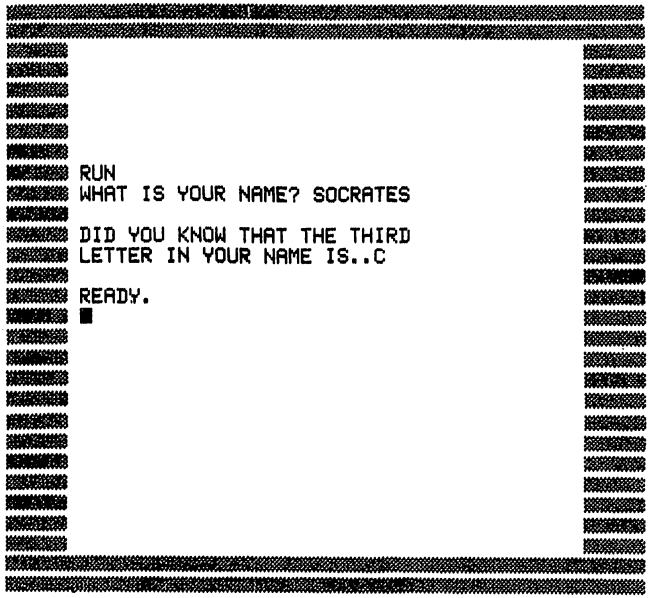

Here is the program to fill out:

```
10 PRINT"GIVE ME A LETTER IN THE ALPHABET"
20 PRINT"AND I'LL TELL YOU THE NEXT LETTER."
30 PRINT"DON'T TRY TO TRICK ME WITH Z BECAUSE"
40 PRINT"I'VE ALREADY LEARNED THAT IT IS THE"
50 PRINT"LAST LETTER."
60 LETA$="ABCDEFGHIJKLMNOPQRSTUVWXYZ"
70 INPUTB$
80 FORX=1TO26
90 PRINT"??????????????????????"
100 NEXT

READY.
```

# 6

# THIRD LETTER

Here is a puzzle that uses the same concept as puzzle 5. You should be able to get this one more easily:

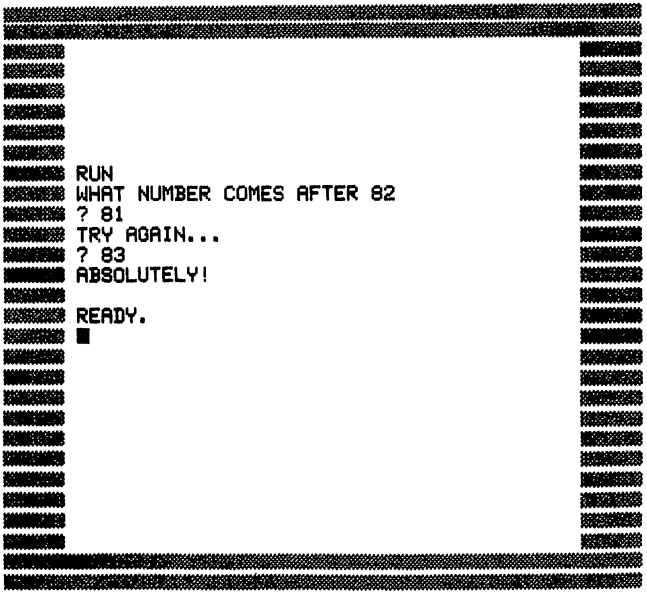



```
10 INPUT"WHAT IS YOUR NAME";A$
20 PRINT
30 PRINT"DID YOU KNOW THAT THE THIRD"
40 PRINT"LETTER IN YOUR NAME IS..";
50 PRINT"??????????????????"

READY.
```

# 7

# NEXT NUMBER

Recently you tried a next letter puzzle. Here is a next number puzzle.

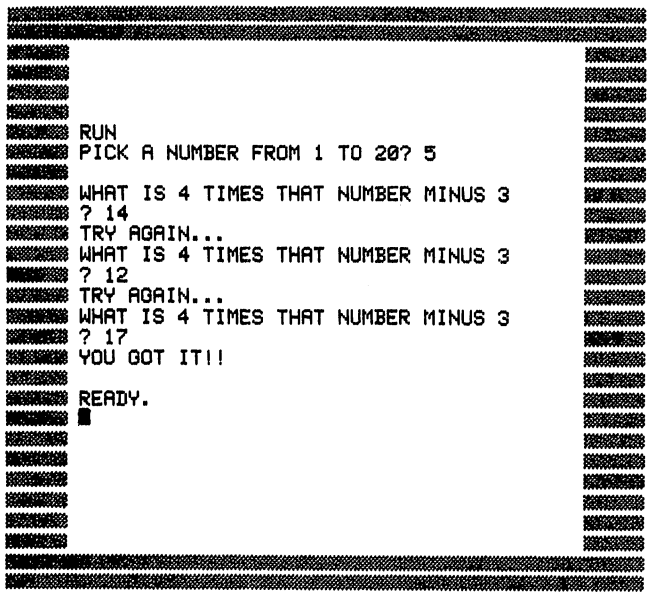



```
10 PRINT"WHAT NUMBER COMES AFTER";
20 LETX=INT(RND(1)*100)+1
30 PRINTX
40 INPUTY
50 IFY=X+1THENPRINT"ABSOLUTELY!":END
60 IFY<>X+1THENPRINT"TRY AGAIN..."
70 PRINT"??????????????????"

READY.
```

# 8

# CALCULATE

Here is a simple arithmetic challenge:




Here is the program for it with a missing line:

```
10 INPUT"PICK A NUMBER FROM 1 TO 20";X
20 PRINT
30 PRINT"WHAT IS 4 TIMES THAT NUMBER MINUS 3"
40 INPUTY
50 PRINT"??????????????????"
60 IFY=ZTHENPRINT"YOU GOT IT!!":END
70 IFY<>ZTHENPRINT"TRY AGAIN..."
80 GOTO30

READY.
```

# 9

# COMPARISON

This program generates two random numbers and asks you which is the larger. It does not give you a second chance but gives you another problem instead.

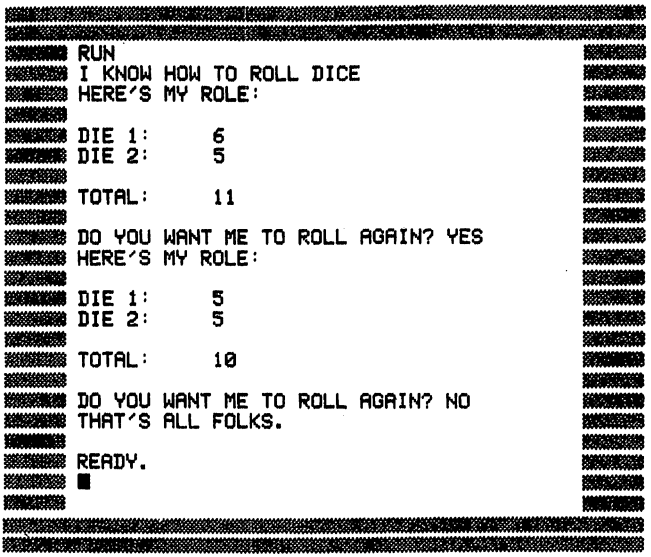
```
RUN
HERE'S A REAL SIMPLE MATH GAME
WHICH NUMBER IS LARGER?

 11        OR          13

? 13
RIGHT, TRY ANOTHER ONE
WHICH NUMBER IS LARGER?

 5         OR          37

? 5
 SORRY, TRY ANOTHER EXAMPLE
WHICH NUMBER IS LARGER?

 28        OR          31

? 31
RIGHT, TRY ANOTHER ONE
WHICH NUMBER IS LARGER?

 28        OR          48

? ■
```

```
10 PRINT"HERE'S A REAL SIMPLE MATH GAME"
20 LET X=INT(RND(1)*50)
30 LETY=INT(RND(1)*50)
40 IFX=YTHEN20
50 PRINT"WHICH NUMBER IS LARGER?"
60 PRINT
70 PRINTX,"OR",Y
80 PRINT
90 INPUTZ
100 IFZ=XTHENGOTO1000
110 PRINT"??????????????????????"
1000 IFX>YTHENPRINT"RIGHT, TRY ANOTHER ONE":GOTO20
1010 PRINT" SORRY, TRY ANOTHER EXAMPLE":GOTO20
2000 IFY>XTHENPRINT"RIGHT, TRY ANOTHER ONE":GOTO20
2010 PRINT" SORRY, TRY ANOTHER EXAMPLE":GOTO20
```

# 10

# DICE

This program sets up the computer to roll a pair of dice and then gives you the total of the two rolls. It can be incorporated into many game programs.




Fill in the missing line:

```
10 PRINT"I KNOW HOW TO ROLL DICE"
20 PRINT"HERE'S MY ROLE:"
30 LETX=INT(RND(1)*6)+1
40 LETY=INT(RND(1)*6)+1
50 PRINT
60 PRINT"DIE 1:",X
70 PRINT"DIE 2:",Y
80 PRINT
90 PRINT"??????????????????????"
100 PRINT
120 INPUT"DO YOU WANT ME TO ROLL AGAIN";A$
130 IFA$="YES"THENGOTO20
140 PRINT"THAT'S ALL FOLKS.":END

READY.
```

# Answers

## 1

```
10 PRINT"WATCH ME COUNT BACKWARDS."
20 INPUT"WHAT NUMBER SHOULD I START FROM";X
30 FORZ=XTO1STEP-1
40 PRINTZ
50 NEXTZ

READY.
```

The missing line decreases the size of X each time the program runs through the loop. You can count backwards by 2's, or 3's or any other number.

## 2

```
10 PRINT"HERE'S AN EASY PATTERN TO PRINT WITH A"
20 PRINT"COMPUTER. TRY TO MAKE YOUR OWN PATTERNS."
30 FORX=1TO750:NEXT
40 PRINT"XXXX XXX XXXX ";
50 PRINT"!!!!!!!!!!!!";
60 GOTO40

READY.
```

It is very easy to make patterns with your Commodore 64. In fact, the last section of this book has a whole series of moderately complex pattern puzzles. In making patterns, you have to look at the arrangement of blank spaces as much as the placement of symbols. The colon is what creates a continuous pattern across the screen and, since the line scrolls around, you will often find patterns that don't look at all like you expect them to. It is a delightful idle activity to explore patterns at random and develop an intuitive sense of how lines like these will look when repeated on the screen in an infinite loop:

```
@@@  ^^^  [[[  ]]]  ^^^  @@@
&&&&&&    %%%%%    &&&&&&
$$$$$ @$@$@$@ &  $$$$$
```

## 3

```
10 INPUT"IS THIS A GOOD DAY FOR YOU";A$
20 PRINT
30 IFA$="YES"THENPRINT"SHARE IT WITH YOUR FRIENDS":END
40 IF A$="NO"THENGOTO100
100 PRINT"MAYBE THIS WILL MAKE YOU FEEL BETTER.."
110 FORX=1TO1500:NEXT
120 PRINTTAB(RND(1)*20)"/";
130 PRINTTAB(RND(1)*20+20)"\"
140 GOTO120

READY.
```

The challenge here is to figure out the way to reconstruct a graphic design from the alphabet of graphic symbols available to you. When you become as familiar with these as you are with the alphabet and numbers you can do interesting graphics without having to PEEK, POKE, or do any complex programming. It is not enough, however, to study the symbols on your keyboard. You have to experiment with them because their power lies in the way they can be combined and not just in the individual symbols.

## 4

```
10 A$="HOPEFUL":B$="INDEPENDENCE":C$="TROUBLE"
20 PRINT"HERE ARE SOME WORDS TO STUDY FOR A FEW"
30 PRINT"SECONDS.  GUESS HOW  MANY  LETTERS  ARE"
40 PRINT"IN EACH WORD."
50 FORX=1TO750:NEXT
100 PRINT
110 PRINTA$
120 INPUT"YOUR GUESS";W
130 IFW=LEN(A$)THENPRINT"GREAT! TRY ANOTHER":FORX=1TO
    750:NEXT:GOTO200
140 IFW<>LEN(A$)THENPRINT"TAKE ANOTHER LOOK":FORX=1TO
    750:NEXT:GOTO100
200 PRINT
210 PRINTB$
220 INPUT"YOUR GUESS";W
    750:NEXT:GOTO300
230 IFW=LEN(B$)THENPRINT"GREAT! TRY ANOTHER":FORX=1TO
240 IFW<>LEN(B$)THENPRINT"TAKE ANOTHER LOOK":FORX=1TO
    750:NEXT:GOTO200
```

```
300 PRINT
310 PRINTC$
320 INPUT"YOUR GUESS";W
330 IFW=LEN(C$)THENPRINT"GREAT! YOU GOT ALL THREE!":END
340 IFW<>LEN(C$)THENPRINT"TAKE ANOTHER LOOK":FORX=1TO
    750:NEXT:GOTO300

READY.
```

This program uses a command you may not be very familiar with, LEN(A$). This command gives you the length of a word or sequence of letters stored in a string. Thus, if there are 50 letters stored in A$, LEN(A$)=50. In this program, the LEN function is used to match your guess against the actual length of the word. LEN is very useful when you are dealing with comparisons of the lengths of strings rather than the specific contents of them.

## 5

```
10 PRINT"GIVE ME A LETTER IN THE ALPHABET"
20 PRINT"AND I'LL TELL YOU THE NEXT LETTER."
30 PRINT"DON'T TRY TO TRICK ME WITH Z BECAUSE"
40 PRINT"I'VE ALREADY LEARNED THAT IT IS THE"
50 PRINT"LAST LETTER."
60 LETA$="ABCDEFGHIJKLMNOPQRSTUVWXYZ"
70 INPUTB$
80 FORX=1TO26
90 IFB$=MID$(A$,X,1)THENPRINT"THE NEXT LETTER IS ";
   MID$(A$,X+1,1)
100 NEXT

READY.
```

The missing line involves using the midstring function MID$(A$,X,Y) that has been used a number of times before in this book. It is another example of how convenient it is to be able to pull a letter or series of letters out of a string whenever you need them in your program.

## 6

```
10 INPUT"WHAT IS YOUR NAME";A$
20 PRINT
30 PRINT"DID YOU KNOW THAT THE THIRD"
40 PRINT"LETTER IN YOUR NAME IS..";
50 PRINTMID$(A$,3,1)

READY.
```

## 7

```
10 PRINT"WHAT NUMBER COMES AFTER";
20 LETX=INT(RND(1)*100)+1
30 PRINTX
40 INPUTY
50 IFY=X+1THENPRINT"ABSOLUTELY!":END
60 IFY<>X+1THENPRINT"TRY AGAIN..."
70 GOTO40

READY.
```

This puzzle deals with setting up a loop to let a person try to correct a wrong answer. The reference of GOTO statements is essential to how a program runs. If you tried

```
70 GOTO 20
```

the target number and, therefore, the correct answer would be changed. It is sometimes useful to deliberately mess up your own programs and see what happens. Sometimes you'll make interesting and useful discoveries. Other times you'll internalize ERROR statements and help yourself create bug-free programs.

## 8

```
10 INPUT"PICK A NUMBER FROM 1 TO 20";X
20 PRINT
30 PRINT"WHAT IS 4 TIMES THAT NUMBER MINUS 3"
```

```
40 INPUTY
50 LET Z=4*X-3
60 IFY=ZTHENPRINT"YOU GOT IT!!":END
70 IFY<>ZTHENPRINT"TRY AGAIN..."
80 GOTO30

READY.
```

In this case, you have to figure out how to perform the calculation asked for correctly and then give its result a new variable name,Z. If you forget to use this new variable you won't be able to use the result of the calculation in other parts of the program without doing a recalculation. In many programs, the introduction of new variables to indicate the results of processes or calculations is a convenient programming tool.

## 9

```
10 PRINT"HERE'S A REAL SIMPLE MATH GAME"
20 LET X=INT(RND(1)*50)
30 LETY=INT(RND(1)*50)
40 IFX=YTHEN20
50 PRINT"WHICH NUMBER IS LARGER?"
60 PRINT
70 PRINTX,"OR",Y
80 PRINT
90 INPUTZ
100 IFZ=XTHENGOTO1000
110 IFZ=YTHENGOTO2000
1000 IFX>YTHENPRINT"RIGHT, TRY ANOTHER ONE":GOTO20
1010 PRINT" SORRY, TRY ANOTHER EXAMPLE":GOTO20
2000 IFY>XTHENPRINT"RIGHT, TRY ANOTHER ONE":GOTO20
2010 PRINT" SORRY, TRY ANOTHER EXAMPLE":GOTO20

READY.
```

This is another puzzle in which you have to figure out the correct GOTO reference. Instead of sending you back to an earlier part of the program as most of the GOTO puzzles did, this one sends you to the end of the program. It is important to realize that the GOTO command can allow you to jump all over your program.

# 10

```
10 PRINT"I KNOW HOW TO ROLL DICE"
20 PRINT"HERE'S MY ROLE:"
30 LETX=INT(RND(1)*6)+1
40 LETY=INT(RND(1)*6)+1
50 PRINT
60 PRINT"DIE 1:",X
70 PRINT"DIE 2:",Y
80 PRINT
90 PRINT"TOTAL:",X+Y
100 PRINT
120 INPUT"DO YOU WANT ME TO ROLL AGAIN";A$
130 IFA$="YES"THENGOTO20
140 PRINT"THAT'S ALL FOLKS.":END

READY.
```

Line 90 lets you both add the total and print it out at the same time. There are times when it is convenient to use the PRINT statement in conjunction with a computation, especially if you do not want to store the result of the computation in any other part of the program.


# SHIFT Graphics Puzzles

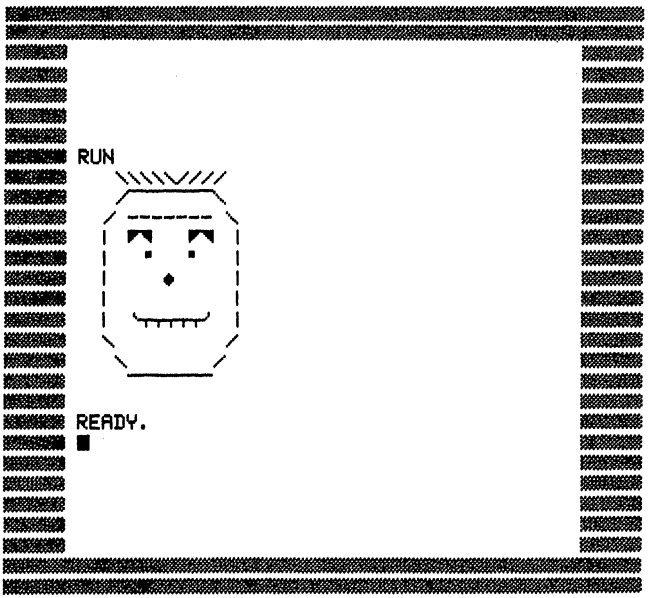
The Commodore 64 computer has two graphics keyboards that are invisible unless you press, the **SHIFT** key or the Commodore Key. (Consult your **Commodore 64 User's Guide** for instructions on using the graphics keyboards.)

The shapes on these keyboards can be used to make very interesting designs and pictures. They can also be mixed with letters to make borders and designs within your ordinary text. They do not need a special graphics command and print straight from the keyboard. They can also be embodied in PRINT statements along with numbers and letters and can be stored in strings the way letters can. There have been examples of these graphics in other parts of the book. Here are a number of puzzles that depend essentially on the use of this graphics mode.

# 1

# A FACE WITH CHARACTER

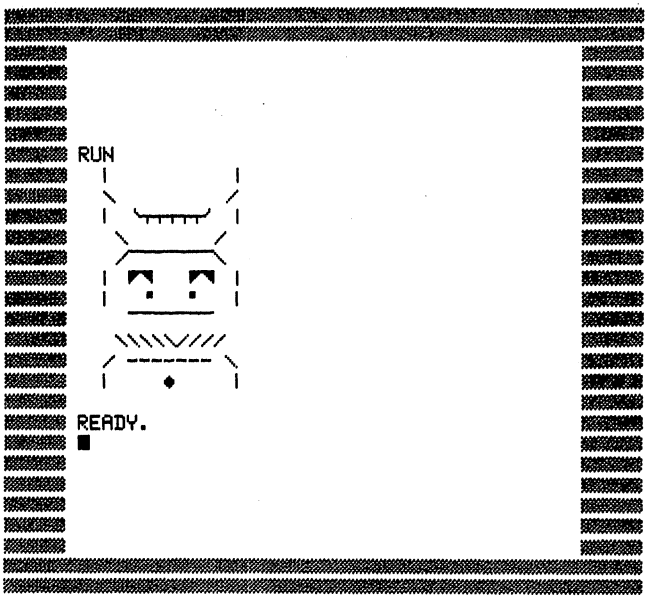
Here is a face, perhaps not the most beautiful one in the world but nevertheless one with character. Using **SHIFT** graphics, try to reconstruct it.



# 2

# PLASTIC SURGERY

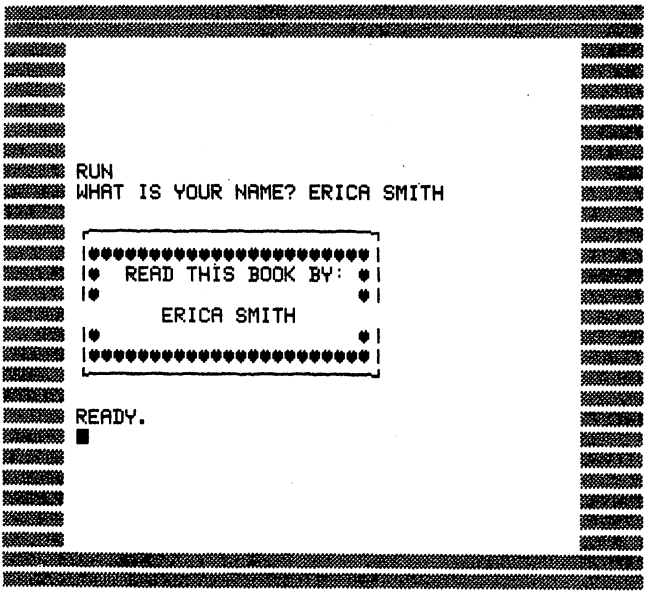
Here's the same face scrambled up as well as the program that generates the scramble. Try to reconstruct the original face by reordering the line numbers without looking at the answers to puzzle 1. The answer to this puzzle is exactly the same as the answer to puzzle 1.



# 3

# TITLE PAGE

This program asks for someone's name and then prints out a simple title page for a book. Construct a program that does this for any name input.

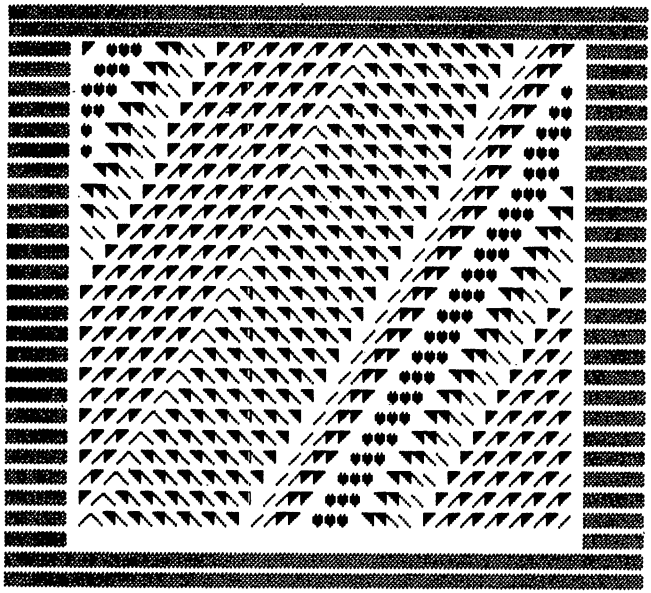


# 4 to 7

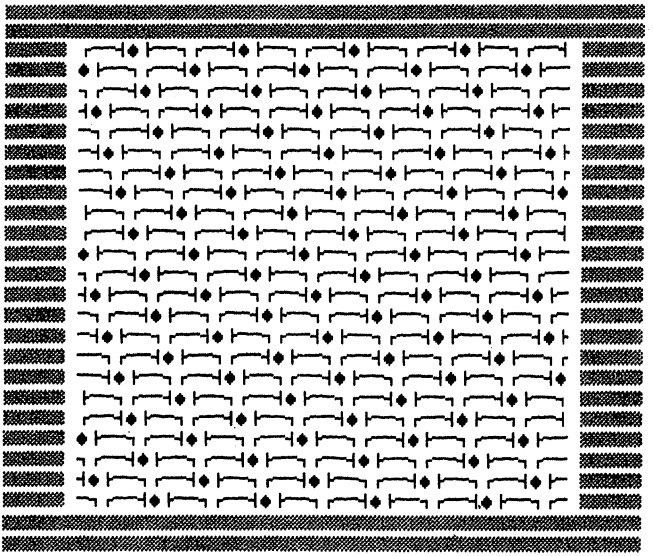
## SHIFT CHARACTER DESIGNS

This is a series of **SHIFT** character designs for you to decipher and try to reproduce. Remember that some of the characters are made up of combinations of other characters. Also, the use of blank spacing is essential to get the design. I suggest you do some sketching and experimenting with combinations of **SHIFT** characters in the course of trying to reproduce these designs. The Appendix contains a sketch pad that is a reproduction of the grid of your Commodore 64 screen. The screen uses regular print as well as **SHIFT** graphics.

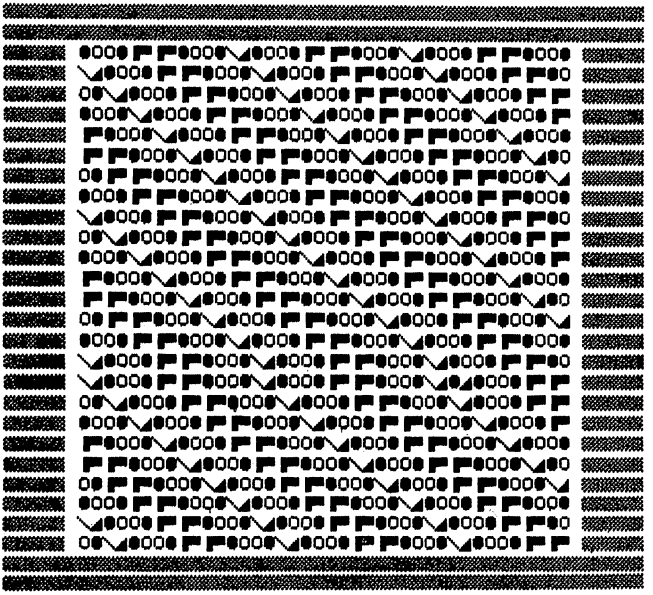
### CLEAR CUTTING IN THE MOUNTAINS:



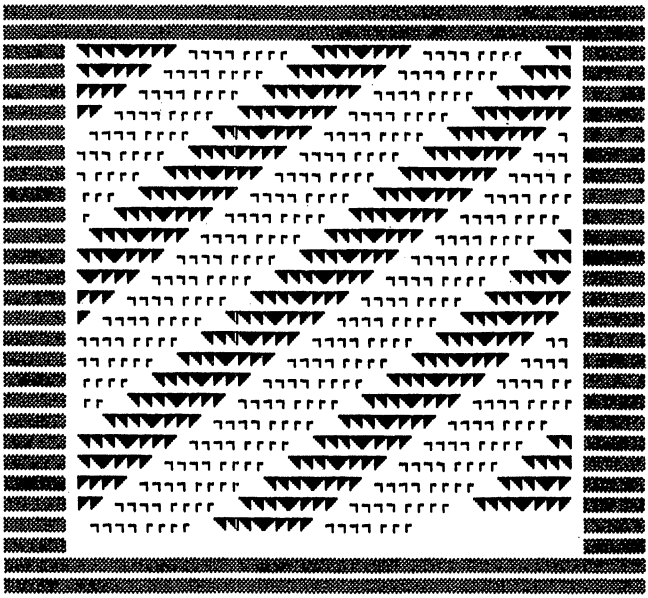
## FENCES AND ROADS:



## DENSE ENVIRONMENT



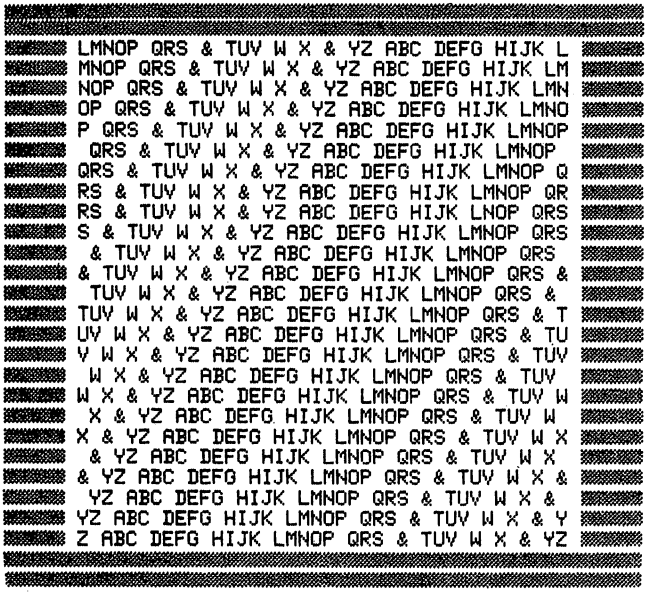
## STRIPES



# 8

# ALPHABET RECONSTRUCTION

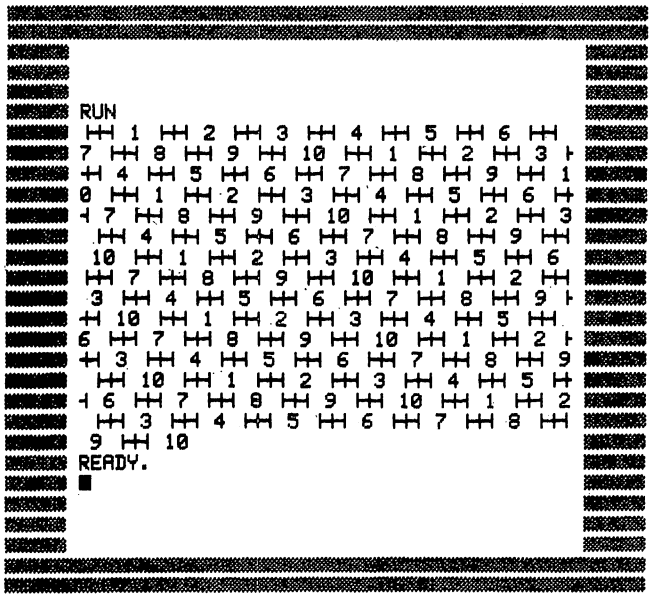
Now that you've spent some time dealing with some fairly complicated designs, here's a simple alphabet to reconstruct. It doesn't use **SHIFT** characters but does use the keyboard and the program structure of BASIC in the same way that the previous programs did.



# 9

# NUMBER GRAPHICS

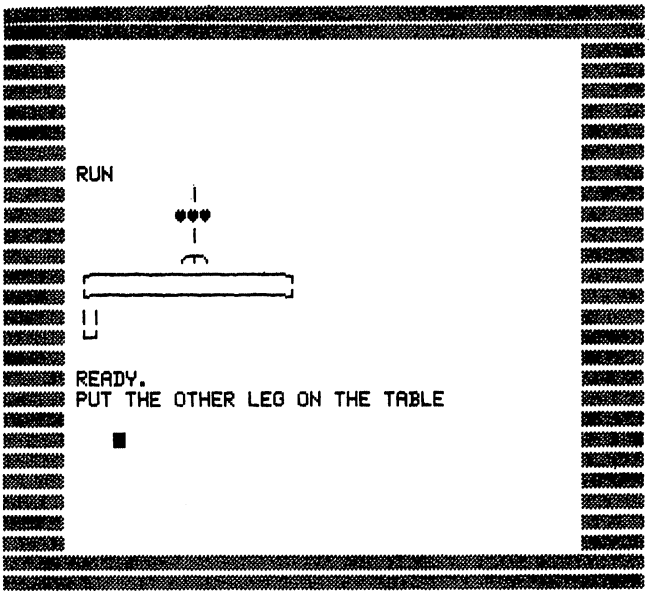
Now that you've had a chance to relax, here's a slightly more complex problem using a mix of **SHIFT** graphics and numbers. Notice that the numbers change in the design in a regular way. The program structure is somewhat different than the ones you've seen before but the program is not much longer.



# 10

# FINISH THE TABLE

Put the missing leg on the table using **SHIFT** graphics. To do it without hints you have to figure out how to draw the whole table. However, if you would like hints here is a program that draws a one-legged table.



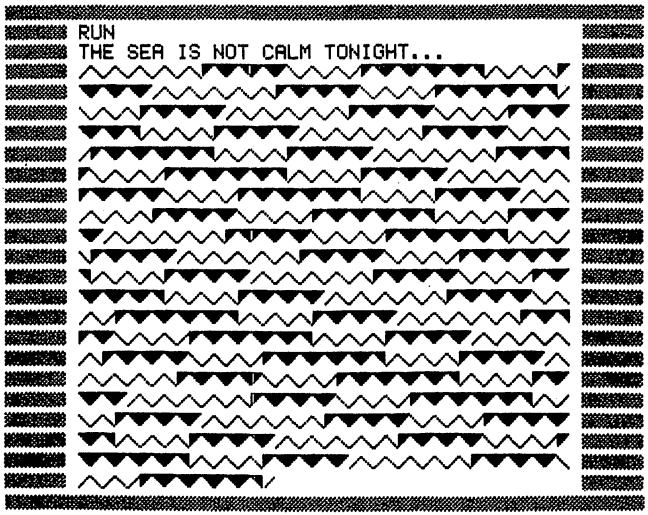

```
10 PRINT"          |"
20 PRINT"         ♣♣♣"
30 PRINT"          |"
40 PRINT"         /-\"
50 PRINT"┌──────────┐"
60 PRINT"└──────────┘"
70 PRINT"||"
80 PRINT"└┘"

READY.
```

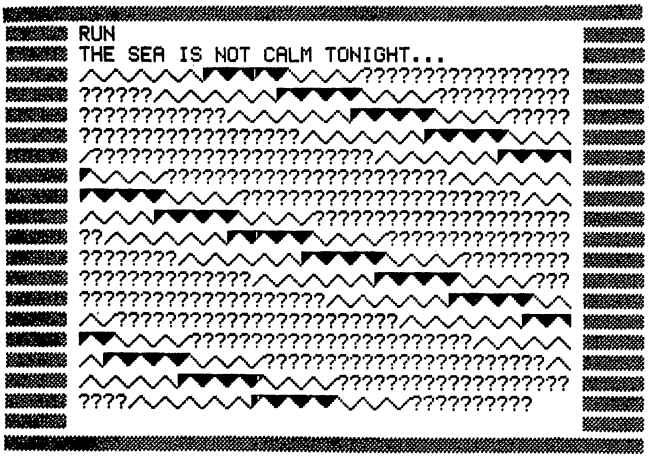
This drawing problem can lead to dozens of others. Try to make some up for your family and friends, and have them be as creative in drawing and challenging you as well.

# 11

# STORMY WEATHER

The sea is not calm tonight, as you can see from the screen dump below. Under that is a puzzling sea, one that questions the very nature of calmness. Can you write programs for each of these screen dumps?



# 12

# SYMBOL ARITHMETIC

Here is a game using numbers and **SHIFT** characters. Three symbols represent numbers. You are asked to memorize the symbol/number equivalents and then do adding using the symbols. It is a good memory game that can be made quite complex. Write a program for this game.

```
RUN
HERE'S A LITTLE SYMBOL-NUMBER GAME:
STUDY THIS TABLE OF VALUES:
     ♠ = 1
     + = 2
     ⊥ = 3

WHEN YOU HAVE THE TABLE MEMORIZED
PRESS 7? ■
```

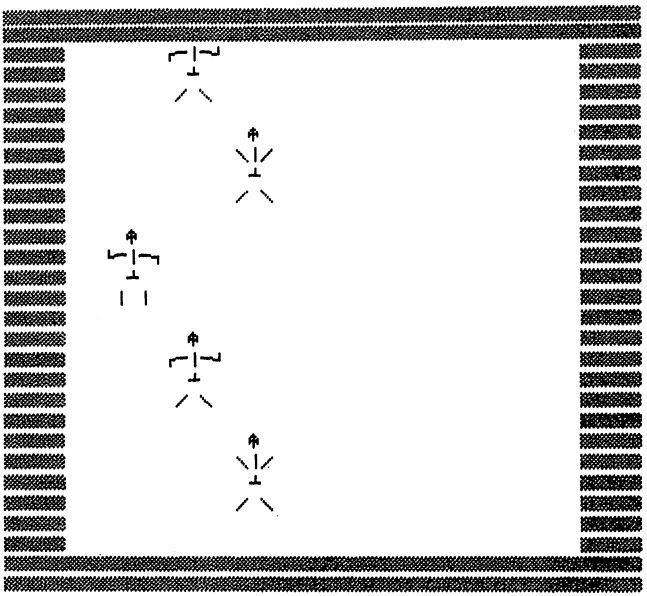
```
HOW MUCH DO THESE ADD UP TO?
♠ ♠ ♠ ♠ ♠ ♠ ♠ ♠ ♠ ? 8
TRY ANOTHER TIME.
? 9
TRY ANOTHER TIME.
? 10
YOU GOT IT DOWN!

READY.
■
```

# 13

# SIMPLE ANIMATION

Here are two simple animations you can make with **SHIFT** graphics. You can actually get quite complex with them although, of course, you won't be able to duplicate the work of Walt Disney or George Lucas. These two dumps show a figure that moves up the screen changing position and posture. See if you can recreate the figures and have them move as in the screen dumps.

**A**



**B**

# Answers

## 1

```
10 PRINT"    \\\\\\////"
20 PRINT"    /      \"
30 PRINT"   /  ------  \"
40 PRINT"   |  ▛  ▜  |"
50 PRINT"   |  ▪  ▪  |"
60 PRINT"   |   ◆    |"
70 PRINT"   |        |"
80 PRINT"   |  ╰────╯|"
90 PRINT"   \        /"
100 PRINT"   \      /"
110 PRINT"    ------"

READY.
```

Notice that the chin seems pushed off the face on lines 100 and 110. When you run the program you'll get the face properly aligned. The reason that these lines are offset is that your Commodore 64 automatically puts a space between the line number and the command following it. Since line numbers 10 to 99 have only two digits they line up. 100 and 110 have three digits so they push the command over one space. You have to back it up in your mind to reconstruct the image the program will run. This is important to remember when you list and edit **SHIFT** character graphics.

## 2

```
10 PRINT"   |        |"
20 PRINT"   \        /"
30 PRINT"   |  ╰────╯|"
40 PRINT"    \      /"
50 PRINT"    /------\"
60 PRINT"   |  ▛  ▜  |"
70 PRINT"   |  ▪  ▪  |"
80 PRINT"    ------"
90 PRINT"    \\\\\\////"
100 PRINT"   /  ------  \"
110 PRINT"   |    ◆    |"

READY.
```

## 3

```
10 INPUT"WHAT IS YOUR NAME";A$
20 PRINT
30 PRINT"┌──────────────────────────┐"
40 PRINT"│♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦│"
50 PRINT"│♦  READ THIS BOOK BY:  ♦│"
60 PRINT"│♦                       ♦│"
70 PRINT"        ";A$
80 PRINT"│♦                       ♦│"
90 PRINT"│♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦♦│"
100 PRINT"└──────────────────────────┘"

READY.
```

## 4 to 7

### 4

```
10 PRINT"◤◢◤◢◤◢◤◢◤◢◤◢◤◥◣◥◣◥◣◥◣◥◣◥◣◥";
20 PRINT" ◢◤◢◤ ♦♦♦ ◥◣◥◣ ";
30 GOTO10

READY.
```

### 5

```
10 PRINT"┌─┤♦├─┐";
20 GOTO10

READY.
```

### 6

```
10 PRINT"▄▟ ●○○●◤◥●○○●▙▄ ";
20 GOTO10

READY.
```

### 7

```
10 PRINT"rrrr  ▼▼▼◆◤◤◤  ┐┐┐┐";
20 GOTO10

READY.
```

## 8

```
10 PRINT"ABC DEFG HIJK LMNOP ";
20 PRINT"QRS & TUV W X & YZ ";
30 GOTO10

READY.
```

Notice that I used three lines in the program instead of two. Using the colon at the end of lines 10 and 20 creates the continuity of the program.

## 9

```
10 FORX=1TO10
20 FORY=1TO10
30 PRINT" ┼┼";Y;
40 NEXTY
50 NEXTX

READY.
```

Notice that the changes in the numbers throughout the program are determined by the nested FOR/NEXT loops. It would be useful if you are not familiar with nested loops to trace the program step by step. For example, the X loop instructs you to print what is inside the Y loop 10 times and then go back to the start and keep on repeating the Y loop 10 full runs. Once you understand this and trace it through you should be able to use nested loops in many different contexts.

## 10

```
10 PRINT"          |"
20 PRINT"        ♦♦♦"
30 PRINT"          |"
40 PRINT"          ⌒"
50 PRINT"┌──────────┐"
60 PRINT"└──────────┘"
70 PRINT"||        ||"
80 PRINT"␣         ␣"

READY.
```

## 11

```
10 PRINT"THE SEA IS NOT CALM TONIGHT..."
20 PRINT"^^^^^^▼▼▼▼^^^^";
30 PRINT"▼▼▼▼▼^^^▼▼▼▼";
40 GOTO20

READY.
```

```
10 PRINT"THE SEA IS NOT CALM TONIGHT..."
20 PRINT"^^^^^^▼▼▼▼^^^^";
30 PRINT"??????????????????????";
40 GOTO20

READY.
```

## 12

```
10 PRINT"HERE'S A LITTLE SYMBOL-NUMBER GAME:"
20 LETA$="✱":LETB$="+":LETC$="⊥"
30 PRINT"STUDY THIS TABLE OF VALUES:"
40 PRINT"      ✱ = 1"
50 PRINT"      + = 2"
60 PRINT"      ⊥ = 3"
70 PRINT
80 PRINT"WHEN YOU HAVE THE TABLE MEMORIZED"
90 INPUT"PRESS 7";X
100 IFX=7THENPRINTCHR$(147)
110 LETY=INT(RND(1)*3)+1
120 LETZ=INT(RND(1)*10)+1
130 PRINT"HOW MUCH DO THESE ADD UP TO?"
140 FROM=1TOZ
150 IFY=1THENPRINT"✱ ";
160 IFY=2THENPRINT"+ ";
170 IFY=3THENPRINT"⊥ ";
180 NEXT
190 INPUTN
200 IFN=Y*ZTHENPRINT"YOU GOT IT DOWN!":END
210 PRINT"TRY ANOTHER TIME.":GOTO190

READY.
```

# 13

```
10 PRINT"     ↑"
20 PRINT"   └─┤─┐"
30 PRINT"     ┴"
40 PRINT"     |  |"
50 PRINT
60 FORX=1TO100:NEXT
70 PRINT"           ↑"
80 PRINT"         ┌─┤─┘"
90 PRINT"           ┴"
100 PRINT"         /  \"
110 PRINT
120 FORX=1TO100:NEXT
130 GOTO10

READY.
```

```
10 PRINT"     ↑"
20 PRINT"   └─┤─┐"
30 PRINT"     ┴"
40 PRINT"     |  |"
50 PRINT
60 FORX=1TO100:NEXT
70 PRINT"           ↑"
80 PRINT"         ┌─┤─┘"
90 PRINT"           ┴"
100 PRINT"         /  \"
110 PRINT
120 FORX=1TO100:NEXT
130 PRINT"                ↑"
140 PRINT"              \ | /"
150 PRINT"                ┴"
160 PRINT"              /  \"
170 PRINT
180 FORX=1TO100:NEXT
190 GOTO10

READY.
```

The key structural elements of these programs could be described as:

PRINT Figure A

PAUSE TO SEE IT

PRINT Figure B

PAUSE TO SEE IT

GO BACK TO FIGURE A

Teasing out the structure of programs in this way can help you create your own programs or modify those other people design.

# Appendix
# Planning Sheets

These pages are designed to help you solve some of the problems in this book. They consist of a series of TV or monitor screens with room under them to write lines of code. They can be used in many different ways and some readers will certainly develop their own aids to solve the puzzles. Here is an example of how they can be used for two puzzle versions of this simple program:

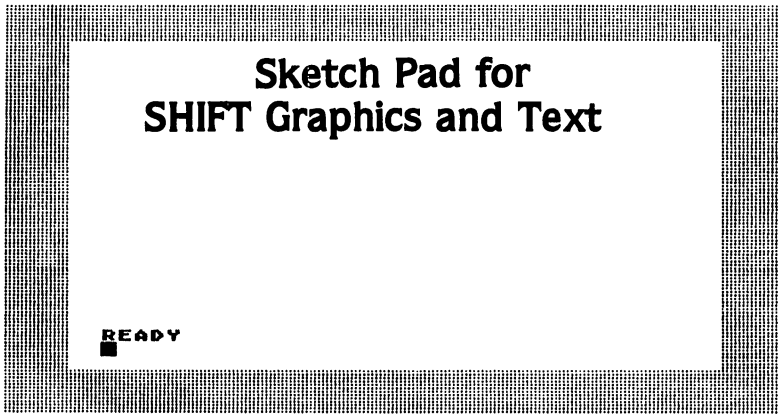
```
10 PRINT "HOW OLD ARE YOU?"
20 INPUT X
30 LET Y=1983-X
40 PRINT "YOU WERE BORN IN ";Y
```

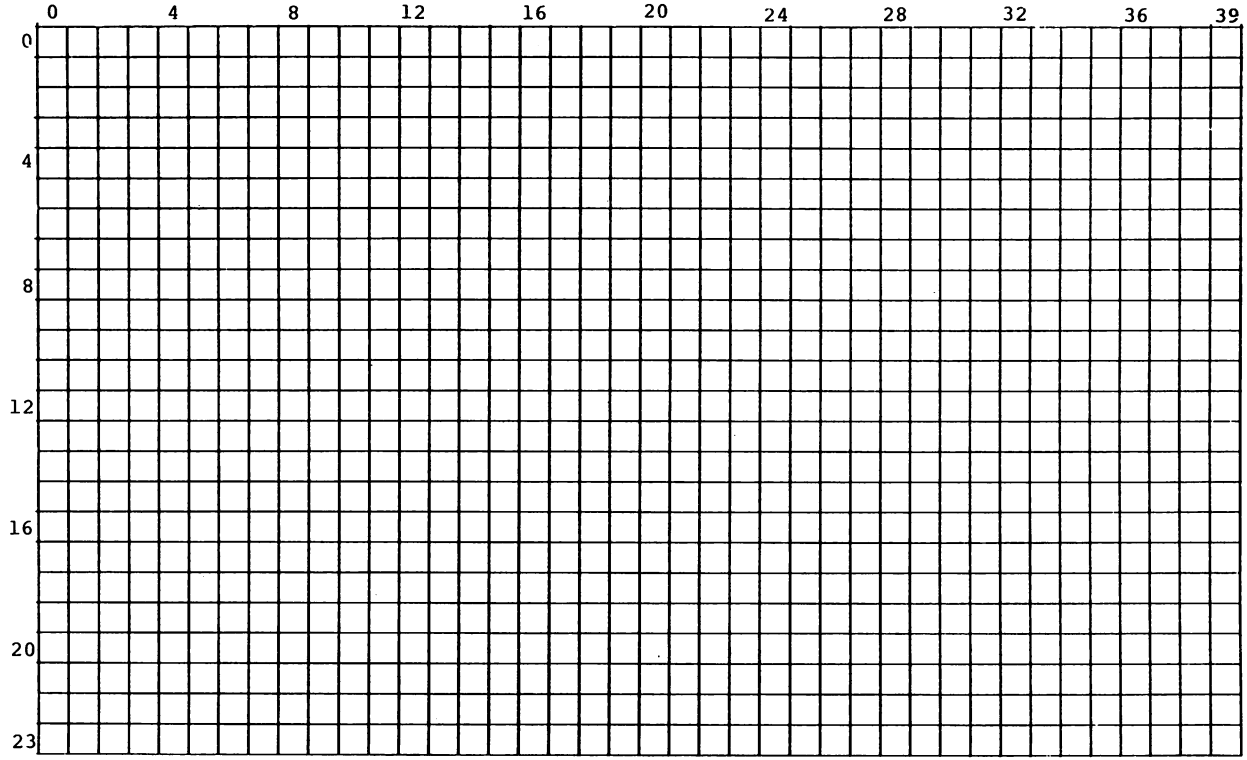
Puzzle version 1:missing line of code

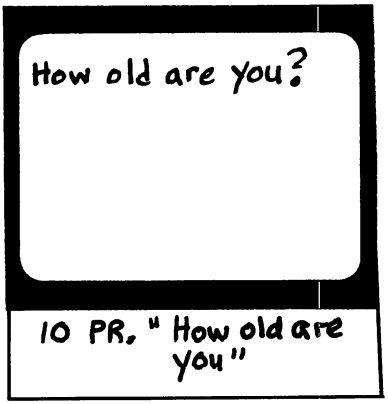
```
10 PRINT "HOW OLD ARE YOU?"
20 INPUT X
30 PRINT "????????????"
40 PRINT "YOU WERE BORN IN ";Y
```

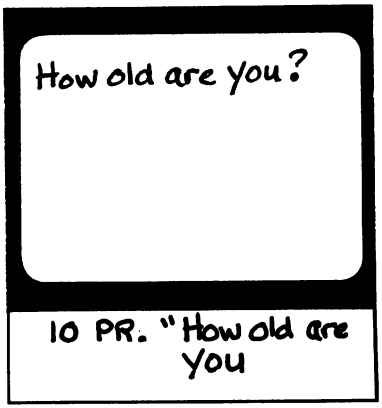
Puzzle version 2:scrambled line number version

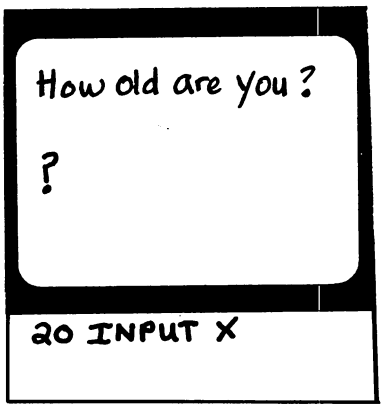
```
10 INPUT X
20 PRINT "HOW OLD ARE YOU?"
30 PRINT "YOU WERE BORN IN ";Y
40 LET Y=1983-X
```

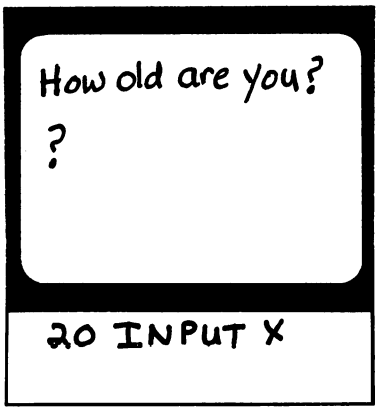

Sketch Pad for
SHIFT Graphics and Text
READY

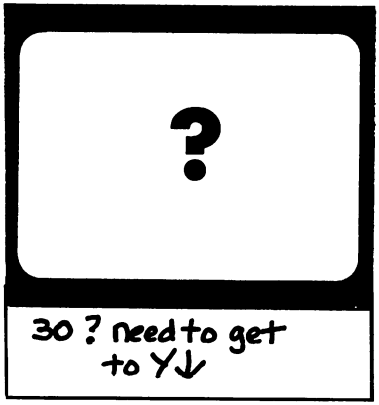

0
4
8
12
16
20
24
28
32
36
39
0
4
8
12
16
20
23

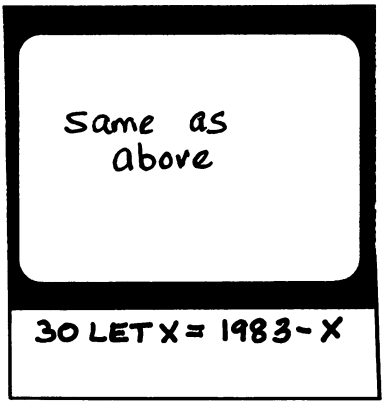

How old are you?
10 PR. " How old are you"

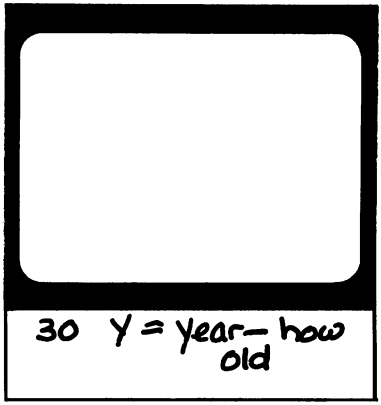

How old are you?
10 PR. "How old are you

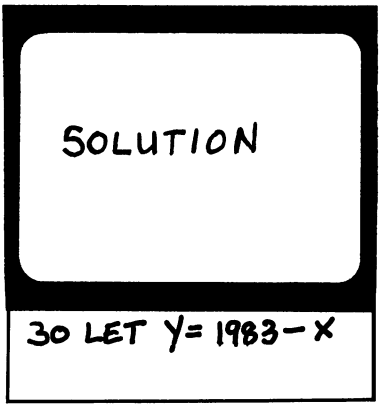

How old are you?
?
20 INPUT X

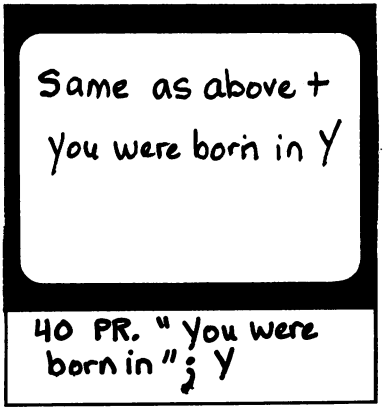

How old are you?
?
20 INPUT X

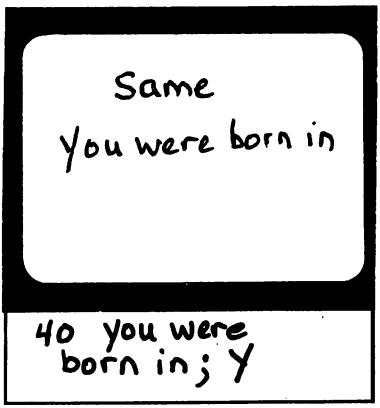

?
30 ? need to get to Y↓

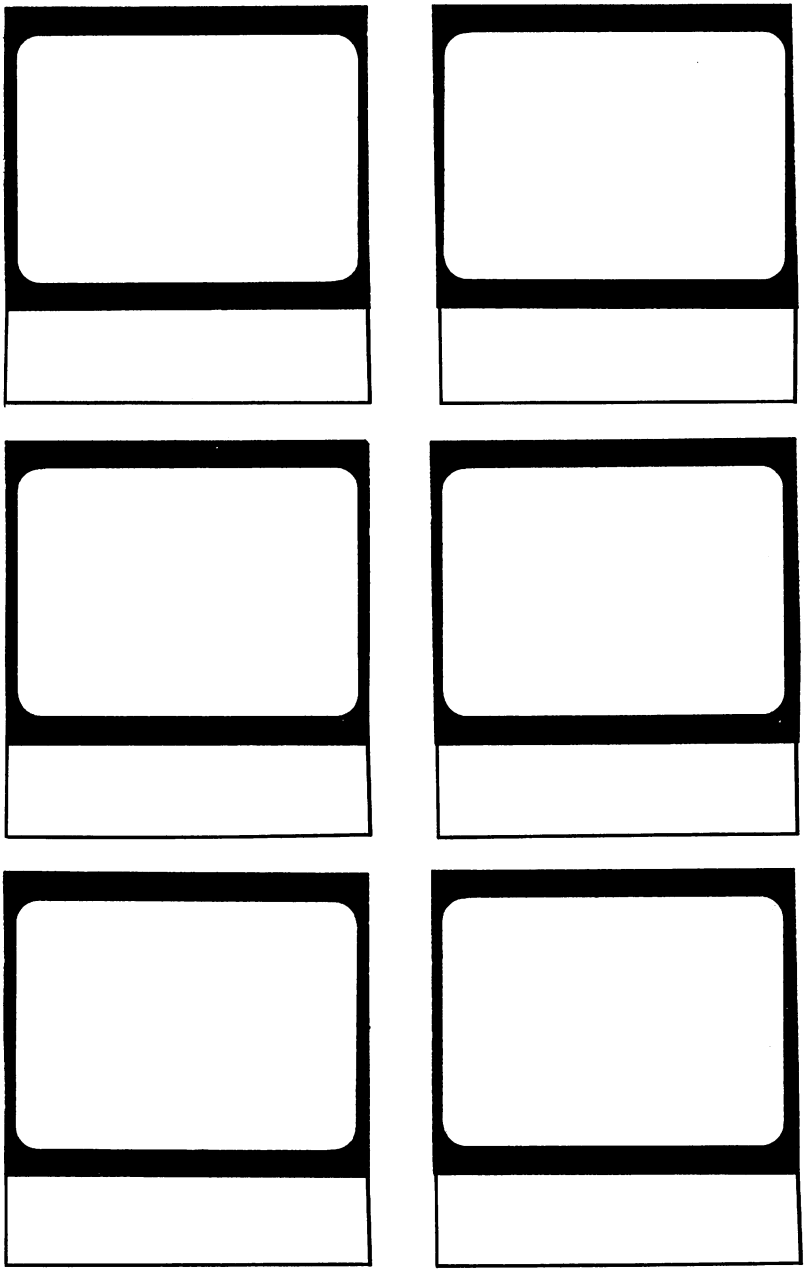

Same as above
30 LET X = 1983 - X

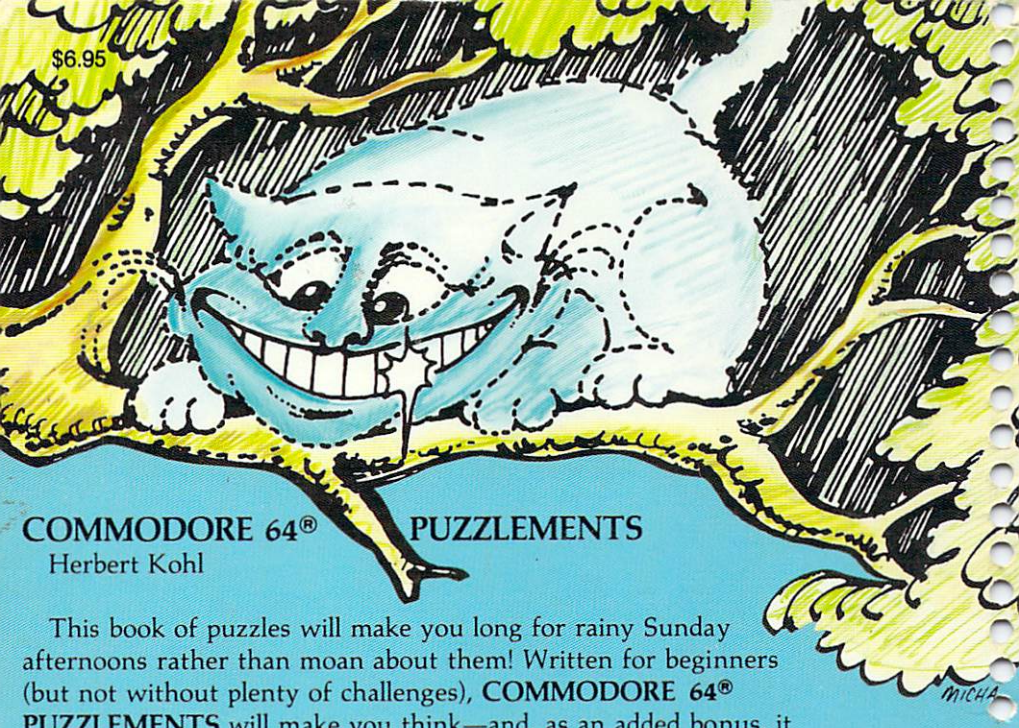

30 Y = year - how old


SOLUTION
30 LET Y = 1983 - X


Same as above +
you were born in Y
40 PR. " you were born in "; Y


Same
you were born in
40 you were born in; Y






# COMMODORE 64® PUZZLEMENTS

Herbert Kohl

This book of puzzles will make you long for rainy Sunday afternoons rather than moan about them! Written for beginners (but not without plenty of challenges), **COMMODORE 64® PUZZLEMENTS** will make you think—and, as an added bonus, it teaches you to think in BASIC. Most of the puzzles can also be worked out on paper, so you can while away the tedium of airplane trips or waiting for the dentist and still sharpen your computer skills.

Mastering the simple programs so elegantly presented for your Commodore Home Computer will provide hours of entertainment for the whole family. Even young readers will enjoy experimenting with combinations of the programs to come up with unique challenges of their own. And, how refreshing to find that your solutions don't have to match an answer key to be exactly, entirely correct.

**A Creative Pastimes Book**
**RESTON PUBLISHING COMPANY, INC.**
*A Prentice-Hall Company*
Reston, Virginia

0 21898 07881 0

illustrations by C. MICHA

0-8359-0788-0