

# A Guide to Programming the Commodore Computers

64 VIC 20<sup>®</sup>

Teacher's  
Guide

BRUCE PRESLEY



Lawrenceville  
Press





# A Guide to Programming the Commodore Computers

BRUCE PRESLEY  
JOHN ELIASON

## TEACHER'S GUIDE

DISTRIBUTED BY



**VAN NOSTRAND REINHOLD COMPANY**

NEW YORK CINCINNATI TORONTO LONDON MELBOURNE

Copyright © 1984  
by



ISBN 0-442-27266-9

All rights reserved. No part of this work covered by the copyright may be reproduced or used in any form or by any means—graphics, electronic, or mechanical, including photocopying, recording, taping, or information storage and retrieval systems—without written permission of the publisher, with the exception of programs, which may be entered, stored and executed in a computer system, but not reprinted for publication or resold in any other form. Printed in the United States of America.



Educational orders may be placed by writing or calling:

Van Nostrand Reinhold Inc.  
7625 Empire Drive  
Attn: Department T  
Florence, Kentucky 41042  
Telephone 606-525-6600

Van Nostrand Reinhold Company  
135 West 50th Street, New York, NY 10020

Macmillan of Canada  
Division of Gage Publishing Limited  
164 Commander Boulevard  
Agincourt, Ontario M1S 3C7, Canada

Van Nostrand Reinhold Australia Pty. Ltd.  
480 Latrobe Street  
Melbourne, Victoria 3000, Australia

Van Nostrand Reinhold Company Ltd.  
Molly Millars Lane  
Wokingham, Berkshire, England RG11 2PY

This guide is written and published by Lawrenceville Press, Inc. and is in no way connected with Commodore Business Machines, Inc. Commodore-64™, VIC-20™, CBM™ and PET® are trademarks of Commodore Business Machines, Inc. with regard to any computer product.

# Acknowledgment

The publication of this teacher's guide represents a major achievement of the staff at Lawrenceville Press in developing curricular materials to assist teachers in presenting an introductory computer programming course. The teaching experience of its authors and the comments of many of the teachers who use our text make us confident that this is the most complete and detailed resource for teachers of programming available.

Many of the lessons, tests and supplementary problems in this guide have been written by John Eliason, a member of the faculty at The Lawrenceville School. An experienced, imaginative teacher, his dedication to teaching is clearly reflected in this guide. A special thanks is also due to Michael Bidwell, Christopher Dingle, Fred Nilson, John Wang and Greg Zaharchuk, who have assisted in every aspect of putting together this guide, and to Jim Adams, who has edited the text to ensure that it is both grammatically and stylistically correct.

Bruce Presley



# Introduction

This Teacher's Guide is a multi-faceted resource that we believe contains all the additional materials you should need to offer your students an excellent introductory programming course.

Each of the chapters in the text has been broken down into a series of lessons intended to be covered in approximately three class periods. Of course, the pace at which the material is covered should be determined by the age and ability of your students. Therefore, the three days per lesson is meant only as a guideline. Maintaining a schedule which is too fast will leave students confused and accomplish little. Since programming is learned sequentially, it is important that students understand each step in the process before going on to the next.

Each of the lessons contains the following features:

**OBJECTIVES:** An outline of the significant topics which should be emphasized.

**ASSIGNMENTS:** A reading assignment is given for the pages in the text which are covered by the lesson. The text problems are the odd-numbered problems whose solutions can be found in the back of the text; these problems are most appropriate for classroom use. The even-numbered problems are used for homework assignments, with the answers to these given in the answer key available only to teachers.

**DISCUSSION TOPICS:** This is additional material which may be used to supplement the text. Its use should be determined by the ability of your class. Often advanced material is presented for those teachers who want to delve into a topic in greater depth than is presented in the text.

**TRANSPARENCY:  
MASTERS** Transparency masters are used in lessons where a program or diagram is the main focus of classroom discussion. The transparency masters may also be photocopied and used as hand-outs.

**WORKSHEETS:** Programming problems which usually do not require the use of the computer are provided. They are especially helpful in classrooms where the number of students exceeds the number of computers.

The following features are found at the end of each chapter in the Guide:

**SUPPLEMENTARY PROBLEMS:** Supplementary problems are provided for whatever use you may have for them. They can be used for homework assignments, classroom discussions or additions to the chapter tests. Answers are given to these problems.

**TESTS:** Tests are provided to determine how well each student has mastered the material in the chapter. Since these tests are designed to emphasize programming skills, most do not contain multiple-choice questions. Again, they should be used selectively, depending on the level of the student; questions can be added or deleted where appropriate. The grading scale provided with the test answers should also be used with discretion. A mid-term examination is given at the end of Chapter 5 and a final examination is at the end of Chapter 9.

Why teach programming?

There are two important reasons for teaching programming which should be kept in mind when preparing an introductory programming course. The first and most important reason is to develop a student's problem solving skills. This is done by requiring students to analyze a problem carefully, break it down into separate parts and, only after working out its details, write the appropriate computer program. Students should not be allowed to sit down at a computer and simply start typing as the first step. This develops sloppy habits and defeats the primary purpose of learning to program. Often it is a good idea to make students present an outline of how they will solve a problem before letting them at the computer. The value to other academic disciplines of learning good problem solving skills can be considerable since what they are really learning is how to think logically. Although the text waits until Chapter Five for a detailed discussion of problem solving, it should be emphasized right from the beginning.

A second reason for teaching students to program is to acquaint them with the capabilities and limitations of computers. Since they are a part of a society which is becoming increasingly computerized, learning to program is probably the best way to make them feel comfortable with computers. Learning only introductory programming is usually enough to convince most students that it is not the computer that is important, but rather the programmer - the human element.

---

Permission is given by Lawrenceville Press, Inc. to teachers using this guide and the publication A Guide to Programming the Commodore Computers to duplicate pages of this teacher's guide only for use in their classrooms.



# Table of Contents

## ACKNOWLEDGMENTS

## INTRODUCTION

### CHAPTER ONE—Introduction to Programming

Lesson 1.1.....	1.1
Worksheet	1.5
Solutions	1.6
Lesson 1.2.....	1.7
Worksheet	1.9
Solutions	1.10
Lesson 1.3.....	1.11
Worksheet	1.15
Solutions	1.16
Lesson 1.4.....	1.17
Worksheet	1.21
Solutions	1.22
Supplementary Problems.....	1.23
Solutions	1.24
Chapter Test.....	1.25
Solutions	1.27

### CHAPTER TWO—Decisions and Loops

Lesson 2.1.....	2.1
Worksheet	2.3
Solutions	2.4
Lesson 2.2.....	2.5
Worksheet	2.9
Solutions	2.11
Lesson 2.3.....	2.13
Worksheet	2.17
Solutions	2.19
Supplementary Problems.....	2.21
Solutions	2.22
Chapter Test.....	2.23
Solutions	2.25

CHAPTER THREE—Computer Games

Lesson 3.1.....	3.1
Worksheet	3.7
Solutions	3.8
Lesson 3.2.....	3.9
Worksheet	3.13
Solutions	3.14
Lesson 3.3.....	3.15
Worksheet	3.21
Solutions	3.22
Supplementary Problems.....	3.23
Solutions	3.25
Chapter Test.....	3.27
Solutions	3.29

CHAPTER FOUR—Nested Loops and Subscripted Variables

Lesson 4.1.....	4.1
Worksheet	4.5
Solutions	4.6
Lesson 4.2.....	4.7
Worksheet	4.11
Solutions	4.12
Lesson 4.3.....	4.13
Worksheet	4.17
Solutions	4.18
Supplementary Problems.....	4.19
Solutions	4.21
Chapter Test.....	4.23
Solutions	4.24

CHAPTER FIVE—Programming Techniques

Lesson 5.1.....	5.1
Worksheet	5.3
Solutions	5.4
Lesson 5.2.....	5.5
Worksheet	5.19
Solutions	5.20
Lesson 5.3.....	5.21
Worksheet	5.23
Solutions	5.24

Supplementary Problems.....	5.25
Solutions	5.28
Chapter Test.....	5.31
Solutions	5.33
Chapters 1 - 5 Examination.....	5.35
Solutions	5.41

#### CHAPTER SIX—Graphics

Lesson 6.1.....	6.1
Worksheet	6.3
Solutions	6.4
Lesson 6.2.....	6.5
Worksheet	6.7
Solutions	6.8
Lesson 6.3.....	6.9
Worksheet	6.11
Solutions	6.12
Supplementary Problems.....	6.13
Solutions	6.14
Chapter Test	6.15
Solutions	6.16

#### CHAPTER SEVEN—Mathematical Functions

Lesson 7.1.....	7.1
Worksheet	7.3
Solutions	7.4
Lesson 7.2.....	7.5
Worksheet	7.7
Solutions	7.8
Lesson 7.3.....	7.9
Worksheet	7.11
Solutions	7.12
Lesson 7.4.....	7.13
Worksheet	7.15
Solutions	7.16
Supplementary Problems.....	7.17
Solutions	7.20
Chapter Test.....	7.23
Solutions	7.25

CHAPTER EIGHT—String Functions and Data Types

Lesson 8.1.....	8.1
Worksheet	8.3
Solutions	8.4
Lesson 8.2.....	8.5
Worksheet	8.7
Solutions	8.8
Supplementary Problems.....	8.9
Solutions	8.10
Chapter Test.....	8.13
Solutions	8.15

CHAPTER NINE—Files

Lesson 9.1.....	9.1
Worksheet	9.5
Solutions	9.6
Lesson 9.2.....	9.7
Worksheet	9.9
Solutions	9.10
Lesson 9.3.....	9.11
Worksheet	9.15
Solutions	9.16
Supplementary Problems.....	9.17
Solutions	9.18
Chapter Test.....	9.21
Solutions	9.23
Final Examination.....	9.25
Solutions	9.31



## FROG

Have the students type FROG and press the RETURN key. Emphasize that the computer will only respond to instructions after the RETURN key is pressed. Slide the opaque screen down to reveal

```
?SYNTAX ERROR
```

and discuss the computer's response.

Continue in this manner to step the students through the transparency. Figure 1 will assist you in your discussions. Invariably, you will need to show students how to correct a typing mistake by using the DEL key.

At the conclusion of the discussion above, provide the students with Worksheet 1 and let them proceed on their own.

NOTE: Remember that before attempting to list a directory of files on a diskette, you must always save whatever program you are currently working on first. This is because the Commodore Disk Directory is stored in a separate file which must be LOAded (LOAD "\$",8) and LISTed to examine its contents.

# Transparency 1

FROG <RET>

?SYNTAX ERROR

NEW

```
10 PRINT "HELLO"  
20 PRINT "GOODBYE"
```

```
RUN  
HELLO  
GOODBYE
```

```
LIST  
10 PRINT "HELLO"  
20 PRINT "GOODBYE"
```

```
SAVE "TRYIT",8
```

NEW

LIST

```
10 PRINT "I LIKE COMPUTING"  
20 PRINT "IT IS EASY"
```

```
RUN  
I LIKE COMPUTING  
IT IS EASY
```

```
LOAD "TRYIT",8
```

```
RUN  
HELLO  
GOODBYE
```

# Explanation of Transparency 1

FROG <RET>	RETURN key signals the computer that our response is complete.
?SYNTAX ERROR	The computer does not recognize FROG as a command.
NEW	Clears the computer's memory.
10 PRINT "HELLO" 20 PRINT "GOODBYE"	A two line program is typed into the computer's memory.
RUN HELLO GOODBYE	The program in memory is translated into instructions the computer can understand. The instructions are carried out in the order of the line numbers.
LIST 10 PRINT "HELLO" 20 PRINT "GOODBYE"	Current contents of memory are listed.
SAVE "TRYIT",8	A copy of your program is stored on diskette as a file named TRYIT.
NEW	Wipes memory clean.
LIST	Current contents of memory contain nothing to list.
10 PRINT "I LIKE COMPUTING" 20 PRINT "IT IS EASY"	A new two line program is typed into the computer's memory.
RUN I LIKE COMPUTING IT IS EASY	The new program is run.
LOAD "TRYIT",8	Finds program named TRYIT on diskette and loads it into memory. The prior contents of memory are erased and lost.
RUN HELLO GOODBYE	Program from disk, now in memory, is run.



# Worksheet 1.1

Part I Type NEW, then enter the given program. RUN the program and see what happens. Then LIST the program to see if it appears the same as when it was typed in. Do not forget to type NEW before each program.

- a) 10 PRINT "MOON"
- b) 10 PRINT "BLUE"
- c) 10 PRINT "FIRST"  
30 PRINT "SECOND"  
20 PRINT "THIRD"
- d) 10 PRINT "TEN"  
20 PRINT "TWENTY"  
10 PRINT "THIRTY"
- e) 10 PRINT "TEN"  
10 10 PRINT "TEN"
- f) 10 PRINT "A"  
20 PRINT "B"  
10
- g) 10 "PRINT WHAT IS WRONG?"
- h) 10 PRINT "SOMETHING IS WRONG"!
- i) 10 PRINT SPIFFY"
- j) 10 PRINT "1+2"  
20 PRINT 1+2

# Worksheet 1.1 Solutions

- Part I Emphasize before students begin that each exercise is to be typed exactly as shown, even if it contains an error.
- a) MOON is printed.
  - b) BLUE is printed.
  - c) The computer executes the commands in line number sequence.
  - d) Reusing a line number erases the line previously given that line number.
  - e) A space inserted in the line number is irrelevant. The computer arranges characters according to an unchangeable convention. So, the second line number is 1010. This can be shown by listing the program with the command LIST.
  - f) Reusing a line number but no command erases the original line.
  - g) Since PRINT is enclosed by the quotation marks the line has no proper command.
  - h) The exclamation point cannot be understood by the computer. Since it lies outside the quotation marks, the computer attempts to interpret it as a command. No such command exists on the computer.
  - i) A quotation mark is missing. (The computer recognizes the imbedded 'IF' as having some special meaning. The value in the variable 'SP' is printed. But don't bring up these ideas now.) Some students may be surprised that inclusion or omission of a single character can cause an error. You might tell your students that in 1962 a single hyphen was missing from a computer program controlling the flight of the Mariner 1 spacecraft. As a result the multi-million dollar craft did not follow the correct trajectory and had to be destroyed.
  - j) Mathematical calculations not enclosed within quotation marks are carried out when the program is RUN, and it is the result of the calculation that is printed.

# Lesson 1.2

- OBJECTIVES:
- a) to recognize how a computer stores information
  - b) to distinguish between a variable and the value assigned to that variable
  - c) to perform simple mathematical calculations on the computer

---

ASSIGNMENT:    Read pages 1.3 through 1.6            Review problems 1 and 2  
                  Text problem 3                            No Homework

---


The concept of variables, dealt with in this lesson, may be the most important lesson in the text. To introduce this idea, multiply two numbers without assigning the result to a variable.

```
10 PRINT 2717 * 599
RUN
1627483
```

Run the program and explain that the result of line 10 is not available for use later on in an expanded program since the result has not been assigned to a variable.

We will visualize the locations where information may be stored in the computer's memory as boxes. Each box will have a unique label, called a variable name. For example, the value


```
10 X = 2717*599                    X            the box is called variable X
20 PRINT X                         1627483       the value assigned to the box
RUN
1627483
```



We say that the value 1627483 has been assigned to the variable X. Emphasize the difference between the name of the box and the value assigned to the box. Remind students that the action of assigning the value to the box does not actually take place until line 10 is executed during a RUN of the program.

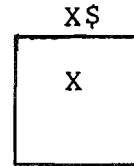
The value of X will be retained in memory until either it is assigned a different value or a new program is run. This is illustrated by adding the following lines to the program above:

```
30 X = 55*384                    X
40 PRINT X                         21120
RUN
1627483
21120
```



Begin another example to illustrate string variables:

```
NEW
10 X$ = "X"
20 PRINT X$
RUN
X
```



The string value "X" is assigned to the variable box X\$. Make sure students understand that X is not used as a variable name. Typically, more than one character is stored in a string variable. The maximum length of a string variable is 255 characters.

Continue the examples as follows:

```
30 X$ = "ABC"
40 PRINT X$
50 PRINT A$
60 PRINT X
RUN
X
ABC
0
```

Students may be confused by use of the variable names X\$ in line 40 and X in line 60. It should be explained that X and X\$ are independent variable names.

Line 60 demonstrates that, even though assigned a non-zero value in the prior demonstration, the variable X now contains the value 0. Numeric variables are set to zero at the beginning of a program RUN and remain zero unless otherwise assigned. Similarly, line 50 shows that string variables are set to null at the beginning of a RUN. When printed, a null does not appear.

Although variable names may be of any length, the text adopts the convention of using only a single letter or a single letter followed by a single digit as a variable name. There are two reasons for this convention. First, only the first two characters are used by the computer to distinguish between variable names. BIG and BID represent the same variable name to the computer. Second, longer variable names may accidentally contain a reserved word. A reserved word is one used as a BASIC command. For example, LONE is not a legitimate variable name because it contains the reserved word ON. There are about 100 reserved words, some of the more common of which are IF, OR, STEP and TO. And they confuse the computer if they are used as part of a variable name.



# Worksheet 1.2 Solutions

Part I The worksheet problems may seem trivial to some students, but point out that the phraseology is significant and promise that a firm grasp of the concept of a variable will avoid numerous problems later. Each problem should be discussed with a keen ear to verbal notation. You may prefer to do the entire worksheet orally. It is highly recommended that you also draw the variable boxes on the chalkboard and show how the values within change during the execution of the program.

- b) 10: The value 5 is assigned to the variable X.  
20: The value 6 is assigned to the variable Y.  
30: The value in box X is multiplied by the value in box Y and the result, 30, is printed.
- c) 10: The value 5 is assigned to the variable X.  
20: The value 6 is assigned to the variable Y.  
30: The value in box X is multiplied by the value in box Y and the result, 30, is assigned to the variable Z.  
40: The value within box Z is printed.

Contrast the techniques of exercises b and c. The technique of c is wasteful unless the value in variable Z is needed later in the program. Encourage students to minimize the number of variables introduced in a program.

- d) 10: The value 17 is assigned to the variable X.  
20: The value in box X, 17, is now assigned to variable Y.  
30: The value within box Y, 17, is printed.
- e) 10: The value 6 is assigned to the variable Y.  
20: The value 8 is assigned to the variable Z.  
30: The values in boxes X, Y and Z are multiplied together and the result, 0, is assigned to box Z. The result is 0 because the value in box X remains 0. All numeric variables are set to 0 at the beginning of program execution and remain 0 unless assigned another value.  
40: The value within box Z, 0, is printed.

## Part II

- a) Since 5 and 6 are typed into line 30, there is no need to save these values within variable boxes.
- b) The quotation marks are not apparently intended in line 40.
- c) Quotation marks are missing around the string in line 10.
- d) Line 30 attempts to print A\$ as a numeric variable A and line 40 to print the numeric variable B as a string B\$.

# Lesson 1.3

- OBJECTIVES:
- a) to recognize three techniques used to assign values to variables: INPUT, READ/DATA and assignment statements
  - b) to learn to judge which assignment statement is most suitable to a given programming problem
  - c) to apply the BASIC commands READ, GOTO and INPUT

---

ASSIGNMENT:    Read pages 1.7 through 1.10    Do review problems 3 and 4  
                  Text problems 5 and 7            Homework problems 4 and 6

---

Three ways are given in Chapter One to change the value within a variable box. The following table provides a summary:

METHOD	EXPLANATION	EXAMPLE
Assignment	Assigns one value to a variable	10 X = 17
READ & DATA	Changes the value of a variable each time the READ is executed	10 READ X 20 GOTO 10 30 DATA 17,3,219
INPUT	Allows the user to change a value while the program is running	10 INPUT X

The three methods may be contrasted using the following program:

```
10 S$ = " IS "  
20 INPUT N$  
30 READ A$  
40 PRINT N$;S$;A$  
50 GOTO 10  
60 DATA (list of adjectives, see below)
```

Allow students to suggest a list of adjectives to be placed in line 60, but do not reveal the structure of the program. When the data list is complete, have the students volunteer to enter their names at the terminal. After the error message (OUT OF DATA) is encountered, LIST and discuss the program. Be sure to point out that it is more efficient to incorporate line 10 directly in line 40.

Some remarks concerning the INPUT statement: Contrast INPUT X\$ with the statement INPUT "WHAT IS YOUR FAVORITE COLOR";X\$. In both cases, a question mark and a blank space are automatically printed.

If a program is to be stopped while it is waiting for INPUT to occur, the user must press the RESTORE key while depressing the STOP key. This operation halts the run of the program, clears the screen and causes the computer to print READY. If the computer is not waiting for input, then just STOP will suspend program execution.

Students sometimes have difficulty recognizing which programming techniques are most useful for solving a given problem. Some examples which may help are presented on the next page. Show each proposed solution and ask the students to explain why it is not the best solution.

PROBLEM 1: Write a program which will calculate the area of a circle from the relation  $AREA=PI * R ^ 2$  where PI is the number 3.1416

```
PROPOSED SOLUTION: 10 INPUT "RADIUS ";R
                   20 INPUT "VALUE OF PI ";P
                   30 PRINT "AREA IS ";P*R*R
```

DISCUSSION: Students should realize that the value PI is a constant (3.1416) and need not be typed in each time the program is run so line 20 should be changed to an assignment statement.

---

PROBLEM 2: Write a program which will print out a list of your friends' telephone numbers.

```
PROPOSED SOLUTION: 10 A$="JACK 111-2222"
                   20 B$="TACK 333-4444"
                   30 C$="MACK 555-6666"
                   40 PRINT A$
                   50 PRINT B$
                   60 PRINT C$
```

DISCUSSION: Assigning a different variable for every data item is unnecessary. A READ/DATA is more appropriate:

```
10 READ A$
20 PRINT A$
30 GOTO 10
40 DATA "JACK 111-2222","TACK
333-4444", "MACK 555-6666"
```

---

PROBLEM 3: A student is working in a lab and needs a program to convert temperature readings from Celsius to Farenheit. The relationship is given by:  
 $F = 9*C/5+32$   
The first reading to be converted is 20 degrees Celsius.

```
PROPOSED SOLUTION: 10 READ C
                   20 F = 9*C/5+32
                   30 PRINT F
                   40 DATA 20
```

DISCUSSION: The experimenter would be greatly inconvenienced if each conversion required retyping RUN X. An INPUT statement would resolve the problem along with a GOTO at line 40.

```
10 INPUT C
20 F = 9*C/5+32
30 PRINT F
40 GOTO 10
```

Note: The use of an infinite loop, which requires that the RUN / STOP key be used, is considered poor programming style and can be avoided using the IF...THEN statement covered in Chapter 2. Also, note that in every case described above the decision was made based on convenience for the program user.



It is wise to place DATA statements at the end of the program because the computer constructs one data list from all available DATA statements. To illustrate, consider the following program:

```
10 DATA 70,20
20 READ X
30 DATA 10,90,30
40 PRINT X/10
50 READ Y,Z
60 PRINT Y * Z
70 DATA 40
80 GOTO 20
```

The program may be simplified to :

```
10 READ X
20 PRINT X/10
30 READ Y,Z
40 PRINT Y * Z
50 GOTO 10
60 DATA 70,20,10,90,30,40
```

Students should be cautioned to make use of GOTO statements only when absolutely necessary. In longer programs they tend to destroy the logical flow of a program and make it much harder to discover errors. It is also considered better programming style to have GOTO statements direct a program back to previous program lines rather than skip lines and jump ahead. For example: 50 GOTO 10 may be acceptable but not 10 GOTO 50.



# Worksheet 1.3

Part I Find the errors, if any, in each of the following programs:

a) 10 READ R  
20 PRINT R \* .06  
30 DATA 17,29,3,6,9

d) 10 DATA 3,9,6,12,15  
20 PRINT S \* .06  
30 READ S  
40 GOTO 10

b) 10 READ A  
20 READ B\$  
30 PRINT B\$;"IS AGE ";A  
40 GOTO 20  
50 DATA 17,JOHN,12,JIM,16,JOAN

e) 10 READ X\$  
20 INPUT X\$  
30 PRINT X\$;" IS A GREEK  
LETTER"  
40 GOTO 10  
50 DATA ALPHA,BETA,GAMMA

c) 10 READ X,Y,Z  
20 PRINT (X + Y + Z) / 3  
30 GOTO 10  
40 DATA 79,67,92,53,99,70,87

f) 10 X = 1  
20 Y = 6  
30 READ X,Y  
40 PRINT X \* Y

Part II Determine the exact output. Be sure to indicate any blank spaces that may occur in the output.

a) 10 READ X  
20 PRINT X \* X  
30 GOTO 10  
40 DATA 1,2,3,4

c) 10 READ A\$,B\$,C\$,  
20 PRINT A\$  
30 PRINT B\$  
40 PRINT C\$  
50 DATA 1 AM,"SNEEZE,PLEASE",,  
KNEES

b) 10 READ X,Y,Z  
20 PRINT X + Y + Z  
30 GOTO 10  
40 DATA 6,2,13,4,9,1,2,3

d) 10 READ B\$,G\$  
20 PRINT B\$;" LOVES ";G\$  
30 READ X,Y,Z  
40 PRINT X;" + ";Y;" = ";Z  
50 DATA DODD,DENISE,MELISSA,2,  
3,5

# Worksheet 1.3 Solutions

## Part I

- a) A GOTO statement is needed in order to evaluate the remainder of the data list.
- b) Line 40 should be GOTO 10.
- c) There are no Y and Z values in the data list to complement 87.
- d) Line 30 must precede line 20. There is nothing technically wrong with line 40, but it is not a good practice to send a program to a data statement. In any case, the data statement usually is typed at the end of the program.
- e) A value is acquired for X\$ in two different ways. One of them is unnecessary or redundant.
- f) Values have already been assigned to X and Y when line 30 is encountered. Line 30 should be deleted.

## Part II

- a) 1  
4  
9  
16  
  
?OUT OF DATA ERROR IN 10
- b) 21  
14  
  
?OUT OF DATA IN 10
- c) 1 AM  
SNEEZE,PLEASE  
  
(The computer interprets a data element between the two commas.)
- d) DODD LOVES DENISE  
  
?SYNTAX ERROR IN 50  
  
(When X is read at line 30, the next item in the data list is MELISSA, but a string cannot be assigned to a numeric variable box.)

# Lesson 1.4

- OBJECTIVES:
- a) to contrast two techniques for the use of the PRINT statement: within a program and immediate mode
  - b) to understand the difference between using a semi-colon and a comma in conjunction with the PRINT statement
  - c) to understand the use of the REM statement
- 

ASSIGNMENT: Read pages 1.10 through 1.15 Do review problems 5 and 6  
Text problems 9,11,13,15 Homework 8,10,12,14

---

The significant part of this lesson is the use of punctuation to control the format of output. For class discussion use Transparency Master 2. Demonstrate the two statements:

```
10 PRINT "A";"B"  
20 PRINT "A","B"
```

Copy the results of printing these lines on the projected grid using each box on the grid to represent the location of a single character. Note that the result of

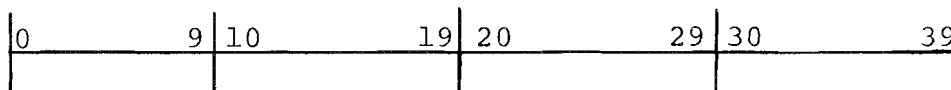
```
30 PRINT "A";  
40 PRINT "B"
```

is identical to the result from line 10 above because of a semicolon at the end of line 30. The semicolon instructs the computer not to perform a linefeed but to continue printing on the same line.

Elaborate on the use of commas by printing the following line:

```
50 PRINT "A","B","C"
```

Illustrate the size of the various comma "zones" by drawing brackets above the grid. It is assumed here that you are working in 40-column format.



Commas are most frequently used in order to create tables. In 40-column format a table created by using commas may have at most 4 columns of information.

PRINT TAB may be introduced as being similar to the tab feature on a typewriter. Pushing the tab key never moves the carriage to the left. Similarly, the computer can only tab from left to right.

Establish a convention for students to use in representing spaces when not working on a grid. The letter b with a slash (b) is often used to represent a blank space.

```
10 PRINT "I HAVEb";X;"bCATS."
```

Caution students that when a problem requests exact output, they must include space symbols where ambiguity may result.

Some of your students may notice that the semicolon is unnecessary at some places within a program. For example, consider the output produced by lines 30 and 40 in the following program:

```
10 X = 5
20 Y = 4
30 PRINT "X=";X;"AND Y=";Y;"!"
40 PRINT "X="X"AND Y="Y"!"
```

```
RUN
X= 5 AND Y= 4 !
X= 5 AND Y= 4 !
```

The outputs are identical. But the statement of line 40 is more obvious. Of course, there is a more profound difference between the statements:

```
PRINT X;Y
PRINT XY
```

The computer considers XY to be the variable name for a single variable.

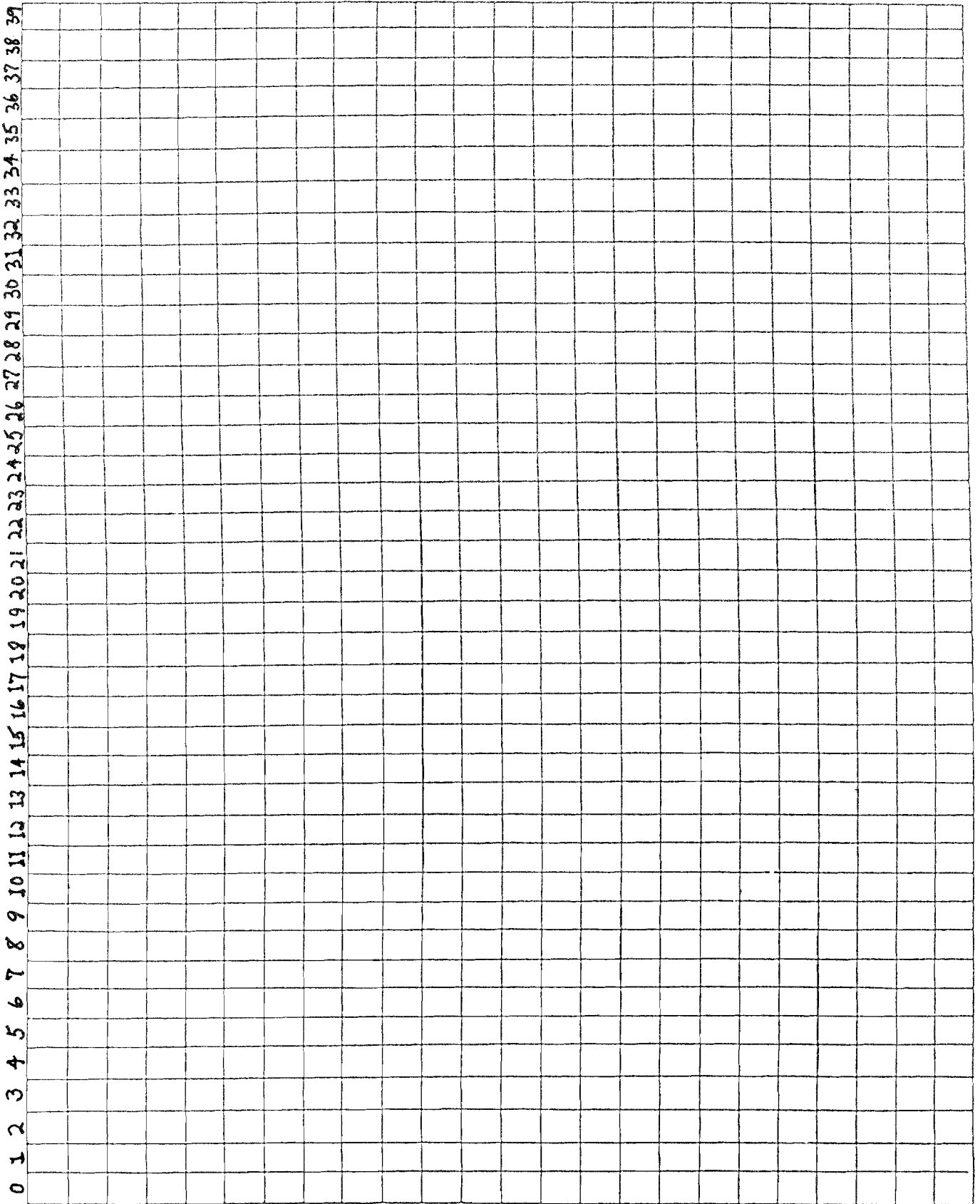
In Lesson 1.2 it was shown that numeric variables are set to zero and string variables are set to null at the beginning of program execution. Any prior contents of the variables boxes are lost. The most recent value is, on the other hand, retained following the end of program execution. Immediate mode can be used to reveal the final value of a variable. For example:

```
10 X$ = "SSSSS...BANG!"
20 N = 5
RUN

PRINT X$;N
SSSSS...BANG! 5
```

Note that the RUN printed nothing since the program contained no PRINT statement, but the values of the variables were printed when immediate mode was used.

# Transparency 1.2









# Worksheet 1.4 Solutions

## Part I

- a) 34
- b) 0

## Part II

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
	1		2			3														
	1									2		3								
										T	O	A	D							
										!										
	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
				*																
	3	0															3	3		
										?	#									
	T	A	B	(	3	)	;	X												

# Chapter 1 Supplementary Problems

- 1) Suppose that material is typed into the computer so that the monitor shows the following:

```
NEW
10 PRINT "RED"
20 PRINT "BLUE"
SAVE "PROG1",8
20
LIST
```

What will now be the output to the monitor? Then, if

```
LOAD "PROG1",8
RUN
```

is entered, what will be output to the monitor?

- 2) Determine the exact output of the following program:

```
10 X = 3
20 Y = 4
30 Z = X + Y
40 Y = Z + X
50 X = Z + Y
60 PRINT X + Y - Z
```

- 3) Write a program which uses the READ statement in conjunction with a DATA statement containing three elements to produce an output similar to the one below. The letters WORD may appear only once in your program.

```
RUN
WORD 1      WORD 2      WORD 3
?OUT OF DATA ERROR IN 10
```

- 4) Write a program that will ask the user for his or her name, and then print this information over and over until interrupted as shown:

```
YOUR NAME?JOHN
JOHN      JOHN      JOHN      JOHN
JOHN      JOHN      JOHN      JOHN
JOHN      JOHN      JOHN      JOHN
JOHN      JOHN      JOHN      JOHN
JOHN      JOHN      JOHN      JOHN
JOHN      JOHN      JOHN      JOHN
JOHN      JOHN      JOHN      JOHN
```

```
BREAK IN 20
```

# Chapter 1 Supplementary Problem Solutions

- 1) a) 10 PRINT "RED"  
b) RED  
BLUE
  
- 2) RUN  
20
  
- 3) 10 READ N  
20 PRINT "WORD";N,  
30 GOTO 10  
40 DATA 1,2,3
  
- 4) 10 INPUT "YOUR NAME";N\$  
20 PRINT N\$,  
30 GOTO 20

# Chapter 1 Test

Part I Determine the exact output of each of the following programs:

- a) 10 X = 10  
20 Y = 12  
30 Z = 8  
40 PRINT X;Y;Z  
50 PRINT X,Y,Z
- b) 10 X\$ = "GOOD SHALL"  
20 Y\$ = "CONQUER ALL"  
30 Z\$ = "!"  
40 PRINT X\$;Y\$,Z\$
- c) 10 Q = 5  
20 R = - 2  
30 S = 3  
40 T = S + (2 \* R) + Q  
50 S = T + R + R  
60 Q = S \* T \* R  
70 PRINT Q  
80 PRINT S  
90 PRINT T
- d) 5 PRINT "MAROON"  
10 X\$ = "RED"  
20 PRINT Y\$  
25 READ X\$,Y\$  
30 DATA PINK,AMBER
- e) 10 PRINT X  
20 X = 3 \* 3  
30 PRINT "X"  
40 PRINT "3\*3"

Part II Circle the programming error(s) within each program given. Explain what must be done to correct the error. If no error exists, state so.

- a) 10 READ X\$  
20 PRINT 2 \* X\$  
30 GOTO 10  
40 DATA 11,22,33
- b) 10 INPUT X;Y  
20 PRINT X \* Y
- c) 10 READ X  
20 PRINT X \* X  
30 PRINT X / X  
40 PRINT X - X  
50 GOTO 20  
60 DATA 2,4,6,8,10
- d) 10 PRINT 2  
20 PRINT 18 - 16  
30 PRINT TWO
- e) 10 READ X  
20 PRINT X  
30 GOTO 10  
40 DATA 1,2,3,4-5

Part III Write each of the following programs as concisely as possible.

- a) Write a program that will allow the user to enter a list of three numbers at the keyboard, then print the average. A sample run will look like:

```
RUN
ENTER 1ST NUMBER? 4
ENTER 2ND NUMBER? 6
ENTER 3RD NUMBER? 11
THE AVERAGE IS 7 .
```

- b) Write a program containing the data statement

```
DATA A,B,C,D,E,F,G,H,I,J,K
```

which will produce the following output:

```
RUN
AB          CD
EF          GH
IJ
?OUT OF DATA ERROR IN 30
```

- c) Write a program which uses INPUT statements to produce an output similar to the one below.

```
RUN
YOUR NAME? JOHN
STATE OF BIRTH? MARYLAND
NUMBER? 90
I'M JOHN, AND I WAS BORN IN MARYLAND.
I HOPE FOR A 90 ON THE FIRST TEST!
```

Don't forget the period after the state name.

# Chapter 1 Test Solutions

The credit for each problem is given in brackets [ ].

Part I [ 5@ = 25 ]

- a)  $\frac{10}{10} \frac{12}{12} \frac{8}{8}$
- b) ~~GOODS SHALL CONQUER ALL~~!!!!
- c)  $\frac{0}{0} \frac{4}{4}$
- d) MAROON
- e)  $\frac{0}{X} \frac{3*3}{3*3}$

Part II [ 5@ = 25 ]

- a) There is a type mismatch in lines 10 and 20. The computer cannot do math operations on a string variable. To correct the error, change X\$ to X.
- b) There is a syntax error in line 10. Replace the semicolon with a comma.
- c) The error occurs in line 50. Line 50 should be as follows:  
50 GOTO 10
- d) The error occurs in line 30. Line 30 should be:  
30 PRINT "TWO"
- e) The error appears in line 40. Mathematical expressions (4-5) are not allowed in DATA statements. Line 40 should be:  
40 DATA 1,2,3,4,5

Part III

- a) 

```
10 INPUT "ENTER 1ST NUMBER";A
20 INPUT "ENTER 2ND NUMBER";B
30 INPUT "ENTER 3RD NUMBER";C
40 PRINT "THE AVERAGE IS";(A + B + C) / 3;"."
```

 [ 15 ]
- b) 

```
10 READ A$,B$
20 PRINT A$;B$,
30 READ C$,D$
40 PRINT C$;D$
50 GOTO 10
60 DATA A,B,C,D,E,F,G,H,I,J,K
```

 [ 20 ]
- c) 

```
10 INPUT "YOUR NAME";N$
20 INPUT "STATE OF BIRTH";S$
30 INPUT "NUMBER";N
40 PRINT "I'M ";N$;"", AND I WAS BORN IN ";S$;"."
50 PRINT "I HOPE FOR A";N;"ON THE FIRST TEST!"
```

 [ 15 ]





# Lesson 2.1

- OBJECTIVES:
- a) to learn the symbols for mathematical operations
  - b) to reduce mathematical expressions which use the computer's order of operations
  - c) to distinguish between conditional and non-conditional statements
  - d) to comprehend the meanings of the conditional symbols and to apply IF...THEN statements in programming applications
- 

ASSIGNMENT:    Read pages 2.1 through 2.6    Do review problems 1 - 4  
                  Text problems 1,3,5,7,9        Homework problems 2,4,6,8

---

Have the students guess the result of the following series of operations:

$$1+3*2^{\uparrow}(2/1+1)$$

Students will disagree. The following table summarizes the convention established for the order of operations on any computer:

- 1) Do operations within parentheses, innermost first.
- 2) Raise to a power ( $\uparrow$ ) from left to right.
- 3) Multiply and divide ( $*$  and  $/$ ) from left to right.
- 4) Add and subtract ( $+$  and  $-$ ) from left to right.

Students should now agree that the above result is 25.

---

The text needs no elaboration with respect to IF...THEN statements. Students generally have no trouble with the idea of a conditional statement or with the conditional symbols. But they will need a great deal of practice in order to become proficient at their application.

It is important to emphasize to students that IF...THEN statements which interrupt sequential program flow should only be used when necessary. Too many of them destroy logical structure and make a program hard to read or debug. If possible, it is best to have an IF...THEN statement send a program back to previous lines rather than to jump over lines to a later point in the program. Obviously this is not always possible.

---

Problems may arise from string comparisons such as:

```
"AB" ? "A "
```

Should the question mark be replaced by '<' or '>'? Point out that the "A " can be considered "A". A space has a "lesser" value than a letter of the alphabet. Therefore:

```
"AB" > "A "
```

---

More advanced students may be directed to the ASCII character code table on page 8.3 of the text. The comparison between two characters is performed according to the value of the decimal code given in the table.

```
"c" > "Z"
```

```
"9" < "A"
```

Therefore, a lower case c is greater than the upper case Z, and a number is smaller than a letter. A space is "less than" the entire set of printable characters.

---

Consider Program 2.3 in the text carefully. Make sure students see the necessity of the second statement of the multiple statement at line 20:

```
IF 2 * X - 18 = 0 THEN PRINT X;"IS THE SOLUTION": GOTO 10
```

The following sample illustrates what would happen if the GOTO 10 is omitted from line 20:

```
RUN
? 9
 9 IS THE SOLUTION
 9 IS NOT THE SOLUTION
?

BREAK IN 10
```

Note that the infinite loop in Program 2.3 is poor programming style and should be avoided wherever possible.

# Worksheet 2.1

Part I Determine the output of each of the following programs:

a)	10 PRINT 3 * 5 + 2 * 5	b)	10 X = 2
	20 PRINT 3 + 5 * 2 + 5		20 Y = 3
	30 PRINT (3 + 5) * (2 + 5)		30 Z = X ↑ Y
	40 PRINT (3 + 5) * 2 + 5		40 W = Y ↑ X
	50 PRINT 4 + 5 + 6 / 3		50 P = Z + W * 2
	60 PRINT 4 + 4 / 2 ↑ 3		60 PRINT Z
	70 END		70 PRINT W
			80 PRINT P
			90 END

Part II Insert '<' or '>' in each pair of strings to create a true expression. Refer to the chart on page 8.3 of the text for help.

a)	"AB"	"BA"	d)	"A"	"AA "
b)	"AB"	"ABC"	e)	"JOHN IS O.K."	"JOHN ISN'T ALL RIGHT"
c)	" A"	"A "	f)	"A "	" Z"

Part III Locate any errors which exist in the following statements:

a) IF X = 1 OR -1 THEN PRINT "THE ABSOLUTE VALUE OF X IS 1"

b) IF X > 0 AND IF Y > 0 THEN PRINT "BOTH VALUES ARE POSITIVE"

c) IF X < 3 AND X > 3 THEN 10

d) IF X = 3 AND Y = 6 THEN IF Z = 4 THEN PRINT "YEAH!"

e) IF X <> 17 PRINT "X DOES NOT EQUAL 17"

Part IV Write the following programs:

a) Write a program in which the user inputs a temperature. Have the program run a test against the temperature to provide the user with a suitable vacation spot. If the temperature is < -32 or > 120, you should print "THIS WEATHER IS UNBEARABLE-STAY AT HOME". If the temperature is between -32 and 40, inclusive, you should print "LET'S GO SKIING". If the temperature is between 80 and 120, inclusive, then the phrase "LET'S GO TO THE BEACH" should be printed. If none of these conditions are met, print "LET'S STAY HOME; IT'S NICE HERE". Use only one PRINT statement. The use of multiple statement lines is suggested.

b) Write a program in which the user will input a value for the variable X. You are to calculate the values for N and M using the formulas below. Print out the larger value using the format "N=..." or "M=...". Try to predict which value will be larger before the run.

$N = X^3 + 3 * X - 10$   
 $M = X^2 + 3 * (X - 10)$

# Worksheet 2.1 Solutions

## Part I

- |    |     |    |    |
|----|-----|----|----|
| a) | 25  | b) | 8  |
|    | 18  |    | 9  |
|    | 56  |    | 26 |
|    | 21  |    |    |
|    | 11  |    |    |
|    | 4.5 |    |    |

## Part II

- |    |   |    |   |
|----|---|----|---|
| a) | < | d) | < |
| b) | < | e) | < |
| c) | < | f) | > |

## Part III

- a) OR -1 should be replaced by OR X = -1
- b) The second IF should be deleted
- c) The statement can never be true
- d) O.K. as it is
- e) THEN is missing before the command PRINT

## Part IV

- a)

```
10 INPUT "ENTER A TEMPERATURE";T
20 IF (T<-32) OR (T>120) THEN A$= "THIS WEATHER IS
   UNBEARABLE-STAY HOME": GOTO 60
30 IF (T>=-32) AND (T<=40) THEN A$ = "LET'S GO SKIING":
   GOTO 60
40 IF (T>=80) AND (T<=120) THEN A$ = "LET'S GO TO THE
   BEACH": GOTO 60
50 A$ = "LET'S STAY HOME; IT'S NICE HERE"
60 PRINT A$
70 END
```
- b)

```
10 INPUT "ENTER A VALUE FOR X";X
20 N = X^3 + 3 * X - 10
30 M = X^2 + 3 * (X - 10)
40 IF N > M THEN PRINT "N ="; N:END
50 PRINT "M =";M
60 END
```

# Lesson 2.2

OBJECTIVE: a) to create repetitive situations using the FOR...NEXT statement

---

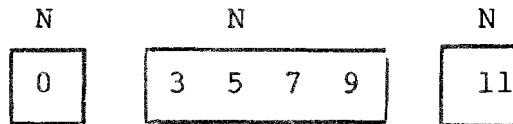
ASSIGNMENT: Read pages 2.7 through 2.12 Do review problems 5 - 7  
Text problems 11,13,15,17,21,23 Homework problems  
10,12,16,22

---

Consider the FOR...NEXT loop:

```
5 PRINT N
10 FOR N = 3 TO 10 STEP 2
20   PRINT N;
30 NEXT N
40 PRINT "LOOP COMPLETE"
50 PRINT N
60 END
```

Using the box analogy, write the list of values the loop variable N may take.



Line 5 proves that the variable is assigned the value 0 before being reassigned at line 10.

Stress that whenever a loop is 'entered', the loop variable must take on the first value in the list. Each time the NEXT statement is encountered, the computer looks back to the list and begins the loop again with N set to the next value in the list. The computer increments the loop variable, then checks to see if the result is beyond the list. In the example the value assigned to N after the loop is completed will be 11.

The importance of indenting the program statements inside of a FOR...NEXT loop cannot be over-emphasized. Type (Commodore) I and add the necessary spaces (we have adopted two as our convention) between the line number and accompanying statement. Indenting makes a program far easier to read and visualize.

In some circumstances a programmer may choose to exit a loop prematurely using an IF statement, but this is usually not advisable and is considered poor programming style. Also, an OUT OF MEMORY ERROR may occur.

The students should be cautioned about tampering with the loop variable while the loop is being executed. Consider the following example:

```
10 FOR N = 1 TO 10
20   PRINT N / 2
30   N = 11
40 NEXT N
```

When NEXT is encountered, N already exceeds the largest value in the list, so the loop is exited.

Conversely, the loop variable may be reset so that the loop continues. This typically results in an endless loop. For example:

```
10 FOR N = 1 TO 10
20   PRINT N / 4
30   N = 3
40 NEXT N
```

The output will be:

```
RUN
.25
1
1
1
1
```

```
BREAK IN 20
```

Altering the loop variable from within the loop is generally an unintentional error generated when a variable is introduced twice in a program for two different purposes. It is one of the first errors to check for when a loop is operating improperly.

If a loop is established in which the initial value of the loop variable is higher than the terminating value, the loop will be executed once. Common sense would seem to dictate that it not be executed at all, but computers do not always operate using common sense. Also, it should be stressed that both the incrementation and testing of the loop variable occur during execution of the ...NEXT statement. This is why the loop will execute at least once and why the number 21 is printed at line 40. For example:

```
10 FOR N = 20 TO 19
20   PRINT N
30 NEXT N
40 PRINT N
```

```
RUN
20
21
```

Students should realize that every time a loop is entered the initial value of the loop variable is reinitialized. For example:

```
10 FOR X = 1 TO 4
20   PRINT "X =";X
30 NEXT X
40 PRINT : PRINT
50 GOTO 10
```

RUN

```
X = 1
X = 2
X = 3
X = 4
```

```
X = 1
X = 2
X = 3
X = 4
```

```
X = 1
```

```
BREAK IN 20
```





## Worksheet 2.2

Part I List the values of the loop variable in the order that they will be assigned within the loop:

- a) FOR N = 1 TO 10 STEP 2                      d) FOR X = 9 TO 6 STEP -2  
b) FOR J = 3 TO 7 STEP 2                      e) FOR Y = 0 TO 99 STEP 33  
c) FOR Z = .5 TO .7 STEP .1                  f) FOR Q = 1 TO 10 STEP 10

Part II Determine the output of each of the following programs:

- a) 10 FOR K = 1 TO 5  
20 PRINT K ↑ 2  
30 NEXT K
- b) 10 X = 3  
20 FOR X = 1 TO 5  
30 PRINT X  
40 NEXT X  
50 PRINT X
- c) 10 FOR N = 1 TO 1  
20 PRINT N  
30 NEXT N
- d) 10 A = 2  
20 FOR X = A TO 3 \* A  
30 PRINT X  
40 NEXT X
- e) 10 FOR J = 1 TO 99  
20 IF J < 39 THEN 50  
30 IF J >= 44 THEN 50  
40 PRINT J  
50 NEXT J  
60 END
- f) 10 X\$ = "ALPHA"  
20 FOR I = 1 TO 5  
30 READ A\$  
40 IF A\$ > X\$ THEN X\$=A\$  
50 NEXT I  
60 PRINT X\$  
70 DATA ALPHA,BETA,GAMMA,  
DELTA,EPSILON
- g) 10 FOR K = 1 TO 3  
20 PRINT K  
30 NEXT K  
40 GOTO 10

Part III Circle the programming error(s) within each program. Explain what must be done to correct the error. If no error exists, state so.

- a) 10 FOR U = 1 TO 20  
20 PRINT U  
30 NEXT V
- b) 10 FOR P = 1 TO 20  
20 PRINT P  
30 GOTO 10
- c) 10 FOR K = 1 TO 3  
20 PRINT K  
30 NEXT K  
40 GOTO 20
- d) 10 FOR S = 9 TO 5  
20 PRINT S  
30 NEXT S



# Worksheet 2.2 Solutions

Part I Note again: the computer increments the loop index and then checks to see if the incremented value is within the range established in the FOR statement.

- |    |           |    |            |
|----|-----------|----|------------|
| a) | 1,3,5,7,9 | d) | 9,7        |
| b) | 3,5,7     | e) | 0,33,66,99 |
| c) | .5,.6,.7  | f) | 1          |

Part II

- |    |    |    |       |
|----|----|----|-------|
| a) | 1  | e) | 39    |
|    | 4  |    | 40    |
|    | 9  |    | 41    |
|    | 16 |    | 42    |
|    | 25 |    | 43    |
| b) | 1  | f) | GAMMA |
|    | 2  |    |       |
|    | 3  | g) | 1     |
|    | 4  |    | 2     |
|    | 5  |    | 3     |
|    | 6  |    | 1     |

Note: Line 10 is of no consequence. Also, a 6 is produced in line 50 because of the final incrementation of the loop variable as discussed on pps. 2.5-2.6.

- |    |   |
|----|---|
| c) | 1 |
| d) | 2 |
|    | 3 |
|    | 4 |
|    | 5 |
|    | 6 |

3
1
2
3
1
2
3
1
(Note: endless loop)

Part III

- a) Line 30 should be NEXT U.
- b) Line 30 should be NEXT P.
- c) Loops cannot be entered without first executing the FOR statement. As it is written, this program will terminate with the error message NEXT WITHOUT FOR ERROR IN 30.
- d) A negative STEP statement is implied.

Part IV

```
10 FOR I = 0 to 18 STEP 2
20 PRINT TAB( I ); "*" ; TAB( 38 - I ); "*"
30 NEXT I
40 FOR I = 18 TO 0 STEP -2
50 PRINT TAB( I ); "*" ; TAB( 38 - I ); "*"
60 NEXT I
```



# Lesson 2.3

## OBJECTIVES:

- a) to apply the concepts contained in Lessons 2.1 and 2.2 in order to solve inequalities and to search data lists
- b) to develop a technique, "input protection", which is used in business and professional programming to protect against improper inputs

## ASSIGNMENT:

Review Chapter 2  
Text problems 29,31      Homework problems 26,32

### A. SOLVING INEQUALITIES

A set of solutions for an inequality may be generated by using a trial and error method. If  $X$  is the variable in question, then the domain of  $X$  is defined by the FOR...NEXT statement. The initial and final values of the FOR...NEXT specify the endpoints of the closed interval over which the inequality will be tested. In text problems of this nature the domain usually contains only integers, but that can be altered using the STEP appendage on the FOR...NEXT statement. For the inequality

$$39X - X^2 > 378$$

the integer solution set is generated by the following program:

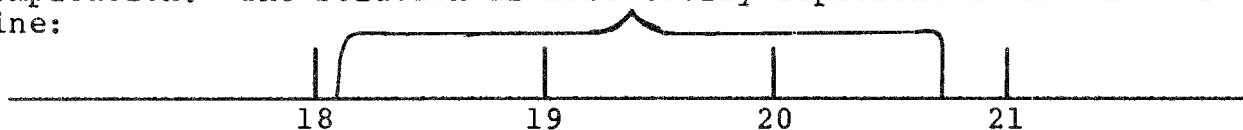
```
10 FOR X = 0 TO 100
20 IF 39 * X - X ^ 2 > 378 THEN PRINT X;
30 NEXT X
RUN
19 20
```

For the integer domain where  $0 < X < 100$ , the program indicates that the only values which satisfy the inequality are 19 and 20. Inspection shows that there could be no solutions outside the chosen domain.

If the domain is not confined to integers, then the endpoints of the solution set may be made more accurate. If the step size is reduced to .1, the endpoints of the solution set will be determined to within .1.

```
10 FOR X = 18 TO 21 STEP .1
20 IF 39 * X - X ^ 2 > 378 THEN PRINT X;
30 NEXT X
RUN
18.1 18.2 18.3 18.4
18.5 18.6 18.7 18.8
18.9 19 19.1 19.2
19.3 19.4 19.5 19.6
19.7 19.8 19.9 20
20.1 20.2 20.3 20.4
20.5 20.6 20.7 20.8
20.9
```

Note that the domain has been narrowed to avoid unnecessary computation. The solution is most easily represented on a number line:



Of course, the endpoints can be determined even more accurately if desired.

---

## B. SEARCHING A DATA LIST

The idea of a "search" to find a specific item of data will be particularly valuable when files are introduced in Chapter 9.

The procedure of searching may be suggested by the commonplace experience of calling "directory assistance" to determine a phone number. The name of the party is entered into a computer, and the computer looks at each item in its data list until a match is found. It knows when a match is found by checking to see if the name entered is the same as the name read from the list. The program for a very small town might look like this:

```
10 INPUT "ENTER NAME";E$
20 FOR I = 1 TO 10
30   READ N$,P$
40   IF E$ = N$ THEN PRINT P$: GOTO 10
50 NEXT I
60 PRINT "NAME NOT ON FILE": GOTO 10
70 DATA GOTWALT,666-1919
71 DATA SMITHERS,666-7890
72 DATA JOHANSON,666-0101
73 DATA BURCKHARDT,666-0009
74 DATA CORSON,666-1111
75 DATA STAWICK,569-1822
76 DATA MACBETH,569-1635
77 DATA LATIMER,666-6846
78 DATA CAMPBELL,659-8989
79 DATA HERTZOG,666-7777
```

Many students will be familiar with the frustration encountered when the exact spelling is not known: E\$ must equal N\$ exactly or the name is NOT ON FILE!

Notice the significance of the multiple statement in line 40. If the GOTO 10 were omitted, the result would be as follows:

```
RUN
ENTER NAME ?MACBETH
569-1635
NAME NOT ON FILE
ENTER NAME ?

BREAK IN 20
```

The RESTORE statement has been purposely omitted from the listing above. In the following RUN see if students can explain why entering CORSON causes an error:

```
RUN
ENTER NAME ?STAWICK
569-1822
ENTER NAME ?CORSON

?OUT OF DATA ERROR IN 40
```

Introduce a RESTORE statement at line 15 and run the program again.

The following program illustrates how to search through data to find the highest valued element:

```
10 FOR I = 1 TO 5
20   READ A$
30   IF A$ > M$ THEN M$ = A$
40 NEXT I
50 PRINT "THE STRING WITH THE HIGHEST VALUE IS ";M$
60 DATA FROG,CAT,ZEBRA,APE,EEL
RUN
THE STRING WITH THE HIGHEST VALUE IS ZEBRA
```

M\$ should be thought of as a variable box which contains the "maximum value encountered so far". At line 30 M\$ is assigned a new string value only when the string value of A\$ is greater than the highest value encountered, M\$. Note that this system works because initially the variable M\$ contains a null string, and a null string is alphabetically less than any character.

Ask your students how the program could be altered in order to find the alphabetically lowest word. Simply reversing the '>' is not sufficient because none of the words will be less than a null string. The solution is to assign M\$ an initial high value.

```
5 M$ = "ZZ"
30 IF A$ < M$ THEN M$ = A$
RUN
THE STRING WITH THE LOWEST VALUE IS APE
```

---

### C. IMPROVING INPUT TECHNIQUES

The student has already seen how the simple statement

```
20 INPUT X
```

can be made more meaningful to the user by writing the following line instead:

```
20 INPUT "ENTER A DATA POINT";X
```

Suppose now that line 20 is to be repeated 100 times, as in the following program:

```
10 FOR I = 1 TO 100
20   INPUT "ENTER A DATA POINT";X
30 NEXT I
```

To facilitate the data entry, it would be helpful to tell the user which data point is about to be entered. A sample RUN would then look like this:

```
RUN
ENTER DATA POINT 1, PLEASE? 7.1
ENTER DATA POINT 2, PLEASE? 6.9
ENTER DATA POINT 3, PLEASE?

BREAK IN 20
```

To accomplish this students will often mix the concepts of PRINT and INPUT:

```
20 INPUT "ENTER DATA POINT";N;" , PLEASE";X
```

The program will perceive the N as a variable to which data is to be assigned. One solution is to write the line:

```
20 PRINT "ENTER DATA POINT";N;" , PLEASE";:INPUT X
```

---

Another technique related to INPUT may be referred to as "input protection". Most professional business programs are protected against improper input. The computer will prevent the user from entering an alphabetic character when a number is expected, but the programmer must supply all other protection. As a simple example, consider a program designed to calculate the percentage grade on a test based upon the number correct out of 210 possible points:

```
10 INPUT "NUMBER CORRECT";C
20 G = C / 210 * 100
30 PRINT "PERCENTAGE GRADE IS ";G
40 GOTO 10
```

The input may be protected here by adding:

```
15 IF C < 0 OR C > 210 THEN PRINT "REENTER": GOTO 10
```

Worksheet 2.3 Part III and Supplementary Problem 6 deal further with the idea of input protection.



# Worksheet 2.3

Part I Determine the output of each of the following programs:

- a) 

```
10 FOR X = - 10 TO 10
20   IF X ↑ 2 < X * 4 THEN PRINT X
30 NEXT X
```
- b) 

```
10 FOR X = - 10 TO 10
20   IF X ↑ 2 > X * 4 THEN PRINT X
30 NEXT X
```
- c) 

```
10 FOR X = - 10 TO 10
20   IF X ↑ 2 = X * 4 THEN PRINT X
30 NEXT X
```
- d) 

```
10 FOR I = 1 TO 5
20   READ A$
30   IF A$ < X$ THEN X$ = "AV"
40 NEXT I
50 PRINT X$
60 DATA APPLE,AJAX,AFTERNOON,AVERAGE,APE
```
- e) 

```
10 FOR A = -2 TO 2
20   C = A ↑ 3 - 3 * A + 3
30   IF C > M THEN M = C:N = A
40 NEXT A
50 PRINT "WHEN A IS";N;"THE MAXIMUM OF";M;"IS ATTAINED."
```

Part II What protection might be added to a program which asks for INPUT of:

- a) item costs in a department store?
- b) numeric month of the year?
- c) hourly salary rate?
- d) current year?

Part III Write each of the following programs as concisely as possible:

- a) Write a program that inputs the coefficients of the quadratic equation and determines whether the roots are real ( $b^2-4ac > 0$ ) or imaginary ( $b^2-4ac < 0$ ).

```
RUN
A,B,C?1,4,4
ROOTS ARE REAL
```

```
RUN
A,B,C?1,1,1
ROOTS ARE IMAGINARY
```

- b) Write a program to search a list of records by artist, and list out all the albums you have recorded by that artist. When the search is completed, have the computer ask if you wish to try again. Check for a "Y/N" type answer. Use lines 110-250 below as DATA statements. Hint: remember to include the RESTORE statement.

```
110 DATA BACH,B MINOR MASS
120 DATA BACH,ORGAN FAVORITES
130 DATA BACH,CONCERTOS
140 DATA MOZART,DON GIOVANNI
150 DATA MOZART,ORGAN MUSIC
160 DATA MOZART,PIANO CONCERTO #26
170 DATA VIVALDI,GREATEST HITS
180 DATA VERDI,FALSTAFF
190 DATA VERDI,OTHELO
200 DATA BEATLES,HEY JUDE
210 DATA BEATLES,REVOLVER
220 DATA BEATLES,SGT.PEPPER
230 DATA JOEL,STRANGER
240 DATA JOEL,TURNSTILES
250 DATA JOEL,COLD SPRING HARBOR
```

```
RUN
ENTER ARTIST'S NAME? JOEL
JOEL      STRANGER
JOEL      TURNSTILES
JOEL      COLD SPRING HARBOR
DO YOU WANT TO CHECK ANOTHER (Y/N)? Y
ENTER ARTIST'S NAME?MOZART
MOZART    DON GIOVANNI
MOZART    ORGAN MUSIC
MOZART    PIANO CONCERTO #26
DO YOU WANT TO CHECK ANOTHER (Y/N)? N
THAT'S ALL FOR NOW...BYE
```

# Worksheet 2.3 Solutions

## Part I

- |    |   |    |     |    |   |
|----|---|----|-----|----|---|
| a) | 1 | b) | -10 | c) | 0 |
|    | 2 |    | -9  |    | 4 |
|    | 3 |    | -8  |    |   |
|    |   |    | -7  |    |   |
|    |   |    | -6  |    |   |
|    |   |    | -5  |    |   |
|    |   |    | -4  |    |   |
|    |   |    | -3  |    |   |
|    |   |    | -2  |    |   |
|    |   |    | -1  |    |   |
|    |   |    | 5   |    |   |
|    |   |    | 6   |    |   |
|    |   |    | 7   |    |   |
|    |   |    | 8   |    |   |
|    |   |    | 9   |    |   |
|    |   |    | 10  |    |   |

- d) None of the words are 'less than' X\$ because X\$ really contains only nulls. The program will produce no apparent output. Show students that adding the line 5 X\$ = "ZZ" will allow the program to search for the word 'closest to the beginning of the dictionary', in this case, AFTERNOON.
- e) WHEN A IS -1 THE MAXIMUM OF 5 IS ATTAINED.

## Part II

- a) No item should have a cost less than or equal to zero.
- b) No value should be less than 1 or greater than 12.
- c) No salary should be less than the minimum wage.
- d) No value less than the year the program was written. Note: an upper limit may also be postulated. The effective usefulness of most programs should be limited to a few hundred years at most. This author knows of certain business programs (written in 1978 by a small company) which will fail after 1999! Not only was this bad programming policy, but the INPUT was not protected against years greater than 1999.

Part III

```
a) 10 INPUT "A,B,C";A,B,C
    20 IF B  $\uparrow$  2 - 4 * A * C < 0 THEN PRINT "ROOTS ARE
        IMAGINARY": END
    30 PRINT "ROOTS ARE REAL"

b) 10 INPUT "ENTER ARTIST'S NAME"; N$
    20 RESTORE
    30 FOR I = 1 TO 15
    40   READ A$,R$
    50   IF A$ = N$ THEN PRINT A$,R$
    60 NEXT I
    70 INPUT "DO YOU WANT TO CHECK ANOTHER (Y/N)"; Z$
    80 IF Z$ < > "Y" AND Z$ < > "N" THEN 70
    90 IF Z$ = "Y" THEN 10
    100 PRINT "THAT'S ALL FOR NOW.. BYE"
    110 DATA BACH,B MINOR MASS
    120 DATA BACH,ORGAN FAVORITES
    130 DATA BACH,CONCERTOS
    140 DATA MOZART,DON GIOVANNI
    150 DATA MOZART,ORGAN MUSIC
    160 DATA MOZART,PIANO CONCERTO #26
    170 DATA VIVALDI,GREATEST HITS
    180 DATA VERDI,FALSTAFF
    190 DATA VERDI,OTHELO
    200 DATA BEATLES,HEY JUDE
    210 DATA BEATLES,REVOLVER
    220 DATA BEATLES,SGT.PEPPER
    230 DATA BILLY JOEL,STRANGER
    240 DATA BILLY JOEL,TURNSTILES
    250 DATA BILLY JOEL,COLD SPRING HARBOR
```

## Chapter 2 Supplementary Problem

- 1) Use the five English words and their Spanish translations and place the ten words in data statements. Display a Spanish word to the user and ask for its English equivalent. Tell the user if the response given is CORRECT or INCORRECT. If the user is incorrect, also tell what the correct response should have been. Continue in this manner until all five translations have been attempted. Use READ...DATA statements to accept your test values.

```
100 DATA HOUSE,CASA
110 DATA EGGS,HUEVOS
120 DATA MILK,LECHE
130 DATA FATHER,PADRE
140 DATA WATER,AGUA
```

```
RUN
WHAT IS THE ENGLISH WORD FOR CASA?HOUSE
CORRECT!!!
WHAT IS THE ENGLISH WORD FOR HUEVOS?EGGS
CORRECT!!!
WHAT IS THE ENGLISH WORD FOR LECHE?FRY
INCORRECT...LECHE MEANS MILK
WHAT IS THE ENGLISH WORD FOR PADRE?WATER
INCORRECT... PADRE MEANS FATHER
WHAT IS THE ENGLISH WORD FOR AGUA?WATER
CORRECT!!!
```

- 2) Write a program which contains twelve adjectives within a data statement. When executed, the program will ask the user for his birth month. The output will be a goofy remark, as shown in the following example:

```
RUN
ENTER YOUR FIRST NAME? JOHN
ENTER YOUR BIRTH MONTH? 11

YOU ARE A VERY FRIENDLY PERSON, JOHN.
```

The secret is to read the data list N times, where N is the month number. In the example above FRIENDLY was the 11th word in the data list. We will find a better method in chapter 6.

- 3) Write a program that will find all integers between 1 and 100, inclusively, that make both of the following inequalities true:  
 $X^3 + X > 50$  &  $X^3 - X < 999$  .

```
RUN
7
8
9
10
```

## Chapter 2 Supplementary Problems Solutions

- 1) 10 FOR X = 1 TO 10  
20 READ A\$,B\$  
30 PRINT "WHAT IS THE ENGLISH WORD FOR ";A\$;: INPUT X\$  
40 IF X\$ = B\$ THEN PRINT "CORRECT!!!": GOTO 60  
50 IF X\$ < > B\$ THEN PRINT "INCORRECT... ";A\$;" MEANS ";B\$  
60 NEXT X  
70 DATA CASA,HOUSE,HUEVOS,EGGS,LECHE,MILK,PADRE,FATHER  
80 DATA AGUA,WATER
- 2) 10 INPUT "ENTER YOUR NAME";N\$  
20 INPUT "ENTER YOUR BIRTH MONTH";N  
30 FOR I = 1 TO N  
40 READ X\$  
50 NEXT I  
60 PRINT "YOU'RE A VERY ";X\$;" PERSON, ";N\$;"."  
70 DATA YOUNG,OLD,HAPPY,HEALTHY,AMBITIOUS,NICE  
80 DATA ATHLETIC,HUNGRY,COOL,NEAT,FRIENDLY,SMART
- 3) 10 FOR X = 1 TO 100  
20 IF (X ↑ 2) + X > 50 AND (X ↑ 3) - X < 999 THEN PRINT X  
30 NEXT X

# Chapter 2 Test

Part I Determine the exact output of each of the following programs:

- a) 

```
10 PRINT 2 + 4 * 3
20 PRINT 1 + 2 + 3 / 2
30 PRINT 20 - 8 / 2 ^ 3
40 PRINT (2 ^ 2 + 2) / 2 + 2
```
- b) 

```
10 FOR I = 1 TO 4
20 READ X$
30 IF X$ < "BA" THEN PRINT X$
40 NEXT I
50 DATA " A", "AB", " B", "B"
```
- c) 

```
10 FOR J = -3 TO 8 STEP 3
20 PRINT J; " ";
30 NEXT J
```
- d) 

```
10 X = 2
20 Y = 3
30 Z = 4
40 FOR X = 1 TO 3
50 Y = X + Z
60 PRINT Y; " ";
70 NEXT X
80 PRINT Z
90 PRINT X
```
- e) 

```
10 FOR X = 2 TO -2
20 PRINT X * X - 1
30 NEXT X
```

Part II Circle the programming error(s) within each program given. Explain what must be done to correct the error. If no error exists, state so.

- a) 

```
10 INPUT X
20 IF X ^ 2 > 16 THEN "X; IS
   A SOLUTION"
30 GOTO 20
```
- b) 

```
10 REM PRINT A LIST FROM 0 TO 9
20 FOR K = 1 TO 10
30 J = K - 1: PRINT J:K = J
40 NEXT K
```
- c) 

```
10 READ P,Q
20 IF P > 0 AND Q > 0 THEN 40
30 IF P < 0 AND Q < 0 THEN 50
40 PRINT "BOTH POSITIVE"
50 PRINT "BOTH NEGATIVE"
60 GOTO 10
70 DATA -1,-3,2,4,5,-17
```
- d) 

```
10 REM PRINT "APPLE" D+1
   TIMES
20 INPUT D
30 FOR I = 1 TO D
40 PRINT "APPLE"
50 NEXT D
60 GOTO 40
```
- e) 

```
10 REM MOUNTAIN STATE
   AREA CODE INDEX
20 INPUT "AREA CODE";A1
30 FOR I = 1 TO 6
40 READ A2,S$
50 IF A1 = A2 THEN PRINT
   "STATE";S$
60 NEXT I
70 PRINT "CODE NOT ON
   FILE"
80 GOTO 20
90 DATA 406,MONTANA,307,
   WYOMING,303,COLORADO,
   208,IDAHO,801,UTAH,
   702,NEVADA
```

Part III Write the following programs as concisely as possible:

- a) Write a program that will ask for the last names of three different people and that will then print out the name that comes first alphabetically. A RUN should look like this:

```
RUN
FIRST NAME ? SMITH
SECOND NAME ? JONES
THIRD NAME ? LYNCH
THE FIRST NAME IS JONES      2.23
```

- b) Write a program which will help young children learn numbers. The program should print the numbers 1 through 10 on the screen, one at a time, and then ask the child to input the correct English spelling of the word. Your program will contain a data statement which looks like the following:

```
DATA ONE,TWO,THREE,FOUR,FIVE,SIX,SEVEN,EIGHT,NINE,TEN
```

Give the child two chances to respond incorrectly, then give the correct answer.

```
RUN
ENTER THE WORD FOR 1?TWO
INCORRECT
ENTER THE WORD FOR 1?THREE
INCORRECT
THE ANSWER IS ONE

ENTER THE WORD FOR 2?TWO
CORRECT (and so on through number 10)
```



# Chapter 2 Test Solutions

The credit for each problem is given in brackets [ ].

Part I [ 5@ = 25 ]

a) RUN  
14  
4.5  
19  
5

d) RUN  
5 6 7 4  
4

b) RUN  
A  
AB  
B  
B

e) RUN  
3

c) RUN  
-3 0 3 6

Part II [ 5@ = 25 ]

a) In line 20 the PRINT command was omitted after the THEN statement. Also the quotation mark preceding 'X;' in line 20 should follow the semicolon. Line 30 should read 30 "GOTO 10" ; otherwise, we would have an endless loop.

b) The last statement in line 30 (K = J) should be omitted. Leaving this statement in would also result in an endless loop.

c) If a positive number were entered, "Both Positive" and "Both Negative" would be printed. To eliminate this problem, rewrite lines 20-50 by inserting

```
20 IF P > 0 AND Q > 0 THEN PRINT "BOTH POSITIVE"  
30 IF P < 0 AND Q < 0 THEN PRINT "BOTH NEGATIVE"  
40 GOTO 10
```

d) Since you can't enter a loop from the middle, line 60 should be changed to:

```
60 PRINT "APPLE"
```

e) Even if the equal condition is met in line 50, the loop will continue to execute and print the "CODE NOT IN FILE" message. Line 50 must be made into a multiple statement line by adding ":GOTO 20" at the end of the line. This will exit you from the loop when the equal condition is met. Also, a RESTORE statement should be added at line 25 to allow the data to be read from the beginning each time the loop is entered.





The value of the argument (X) of the RND(X) statement may be any integer. The value of the argument is irrelevant. It is rather the sign of the argument ( -,0,+ ) which decides the outcome of the RND statement.

---

It should be noted that a program designed to print a list of random numbers using a positive integer as an argument will produce the same set each time the program is first run after the computer is turned on. This creates a serious problem when running game programs or any program where the results should not be predictable. For example, typing in the following program again and running it will yield identical numbers:

```
10 FOR I = 1 TO 4
20   N = RND (1)
30   PRINT N
40 NEXT I
```

```
RUN
.973136996
.103117626
.0277148333
.779343355
```

(Turn off the computer and re-enter the program and run it.)

```
RUN
.973136996
.103117626
.0277148333
.779343355
```

To guarantee a new set of random numbers each time a program is run, the argument (X) should be set to zero. A new set of random numbers will be produced for subsequent random statements each time the computer is turned on. Therefore, an argument (0) should be used for most situations in which random numbers are used. For example:

```
10 FOR I = 1 TO 4
20   N = RND (0)
30   PRINT N
40 NEXT I
```

```
RUN
.365698131
.830384226
.947350239
.131733636
```

(Turn off the computer and re-enter the program and run it.)

```
RUN
.968498391
.584208112
.760921408
.183321568
```

However, it is sometimes useful to have a set of "random" numbers which is predictable. One such instance would be in the debugging of a program. If you are attempting to fix an error in a program, using the same data on each RUN is enormously helpful. This is accomplished by the use of a negative argument in the RND statement. Although the same results can come about from using a positive argument and turning the computer off and on, the negative argument allows for predictable outcome without re-entering the program. For example:

```
10 X = RND(-1)
20 FOR I = 1 TO 5
30   PRINT RND(X)
40 NEXT I
```

```
RUN
.328780872
.978964086
.895758909
.161031701
.0224078245
```

```
RUN
.328780872
.978964086
.895758909
.161031701
.0224078245
```

Line 10 produces the initial "seed" for the RND statement in line 30. It should be noted that a negative argument only produces an initial "seed" for subsequent RND statements to act upon. Also, each negative number produces a unique "seed". For example:

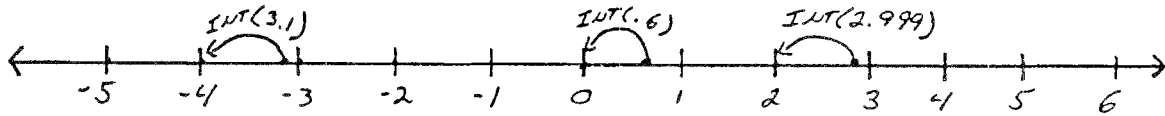
```
10 FOR X = -10 TO -1
20   PRINT X,RND(X)
30 NEXT X
```

```
RUN
-10      3.73729563E-08
-9       3.3647666E-08
-8       2.99223757E-08
-7       5.2273208E-08
-6       4.48226274E-08
-5       3.73720468E-08
-4       2.99214662E-08
-3       4.48217179E-08
-2       2.99205567E-08
-1       2.99196472E-08
```

Students maybe somewhat perplexed that

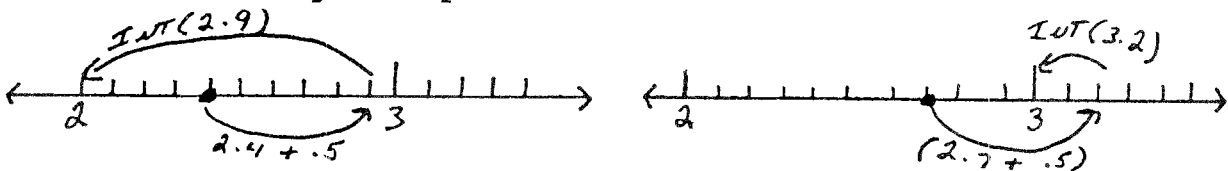
```
PRINT INT(-3.1)
```

does not produce the output -3. To show why, use a number line analogy. First show several examples in the positive range, then discuss -3.1.



The INT statement always generates the integer found to the left of a non-integer argument on a traditional number line.

The number line is also useful for discussions concerning the process of rounding. Compare:



While developing the formula for generation of a set of integers between a and b, inclusive, allow the students to see where '+1' in

$$B - A + 1$$

comes from. At a glance many students will say that there are 10 elements in the set

$$15, 16, 17, \dots, 25$$

but when the set is written out as

$$15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25$$

it is easy to see that the set contains one more than the difference between the end points.

The ability to determine whether a number is even or odd is a useful tool. You may prefer to let students develop the strategy needed on their own. If you decide instead to discuss this application in class, the following program provides a useful summary:

```
10 INPUT "ENTER ANY INTEGER";N
20 IF N <> INT (N) THEN PRINT "NOT AN INTEGER":GOTO 10
30 IF N / 2 = INT (N / 2) THEN PRINT "EVEN"
40 IF N / 3 = INT (N / 3) THEN PRINT "MULTIPLE OF 3"
```

Using a strategy similar to that in lines 30 and 40, your students should now be able to write a program to test a number to determine whether it is prime. A prime number is a number which is evenly divisible only by itself and 1. The following program is the one your students should produce:

```
10 INPUT "INTEGER > 2 PLEASE";N
20 FOR X = 2 TO N - 1
30   IF N / X = INT (N / X) THEN 70
40 NEXT X
50 PRINT "THAT INTEGER IS PRIME."
60 END
70 PRINT "THAT INTEGER IS NOT PRIME."
```





# Worksheet 3.1

Part I For each range specified below write a single statement which will generate one random integer within the range:

- a) 0 and 100, inclusive
- b) 1 and 100, inclusive
- c) 0 and 1000, inclusive
- d) 1 and 2, inclusive
- e) 17 and 24, inclusive
- f) -3 and 3, inclusive
- g) 0 and -100, inclusive

Part II Determine the output of each of the following programs:

- a)

```
10 FOR I = 1 TO 6
20   READ X
30   PRINT INT (X)
40 NEXT I
50 DATA 2,2.001,1.999,-2,-2.001,-2.999
```
- b)

```
10 FOR I = 1 TO 10
20   READ X
30   PRINT INT (X + .5)
40 NEXT I
50 DATA .001,.499,5.00,5.01,5.99,-.001,-.499,-5.00,
-5.01,-5.99
```

Part III Write the following program as concisely as possible:

- a) Write a program to simulate a pair of dice and have the computer print the following result. Any other input should terminate the program.

```
RUN
8
HIT RETURN TO ROLL AGAIN?

11
HIT RETURN TO ROLL AGAIN?

9
HIT RETURN TO ROLL AGAIN?

7
HIT RETURN TO ROLL AGAIN?
```

# Worksheet 3.1 Solutions

## Part I

- a) PRINT INT(RND(0)\*101)
- b) PRINT INT(RND(0)\*100+1)
- c) PRINT INT(RND(0)\*1001)
- d) PRINT INT(RND(0)\*2+1)
- e) PRINT INT(RND(0)\*8+17)
- f) PRINT INT(RND(0)\*7-3)
- g) PRINT INT(RND(0)\*101-100) or PRINT INT(RND(0)\*(-101))

## Part II

- a) 2  
2  
1  
-2  
-3  
-3
- b) 0  
0  
5  
5  
6  
0  
0  
-5  
-5  
-6

## Part III

- a) 10 PRINT INT ( RND (0) \* 6 + 1) + INT ( RND (0) \* 6 + 1)  
20 INPUT "HIT RETURN TO ROLL AGAIN";D\$  
30 IF D\$ = "" THEN 10  
40 END

## Lesson 3.2

OBJECTIVES: a) to comprehend the technique of summation  
b) to apply the summation technique to counting and accumulating applications

---

ASSIGNMENT: Read SUMMATION section Do review problem 3  
beginning on page 3.3  
Text problems 7,15,17,19 Homework problems 6,10,18,26

---

To introduce the idea of summation, first demonstrate a simple counter. A counter will be a variable to which we add 1 repetitively, as is done in line 10 of the following program:

```
10 S = S + 1
20 PRINT S
30 GOTO 10
```

Line 10 must be explained carefully. The box analogy will prove to be very helpful here, but begin by considering a seemingly unrelated example:

```
10 Y = 5
20 X = Y + 1
30 PRINT X
```

At line 20,  $Y + 1$  must be evaluated: the computer finds the current value of  $Y$  and adds 1 to it. The result is then assigned to the variable  $X$ .

Now, in the case at hand, we note first that  $S$  is initially zero:

S  

0
---

When line 20 is executed,  $S + 1$  must be evaluated: the computer takes the current value of  $S$ , 0, and adds 1 to it. The result, 1, is now assigned to the variable  $S$ . (In practice erase 0 and insert 1 in the box.)

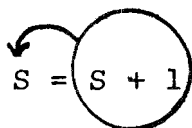
S  

1
---

The 0 present in  $S$  is now erased and replaced by 1. Repeat the process two or three times for emphasis. The operation can be further stressed by drawing

$S = \textcircled{S + 1}$

to indicate that the right side is evaluated using the 'old' value of S, then



to show that the result is assigned 'back to' the variable S.

In game applications summation is frequently used to keep score.

The simple counter is just one of the several techniques which may be grouped under the heading 'summation'. Your students should now be able to predict the output of

```
10 N = 4
20 FOR X = 1 TO 4
30   S = S + N
40   PRINT S
50 NEXT X
RUN
4
8
12
16
```

The value of N in line 10 selects the 'step size' for counting and so this program might be called a 'count by N' counter.

A still more useful form of summation can be referred to as an 'accumulator'. Consider the following program:

```
10 FOR I = 1 TO 5
20   INPUT N
30   S = S + N
40 NEXT I
50 PRINT "SUM=";S
RUN
? 4
? 2
? 6
? 9
? 1
SUM= 22
```

This program allows the five values entered at line 20 to be accumulated or summed in variable box S. In this sense, S may be called an accumulator. The contents of the accumulator (the sum of the 5 numbers) are ultimately printed out at line 50.

---

Progressions provide an interesting application of the summation technique. Bank interest and population growth problems are of this type. Do not venture into such problems until your students are comfortable with summation.

In compound interest problems the value to be added to the accumulator depends on the most recent value of the accumulator. The interest to be added to the balance (at the end of each compounding period) is a fixed percentage of the balance.

For example, if \$1,000 is invested in a ten year bond which earns 8% compound interest each year, the following program will produce the yield at the end of 10 years.

```
10 S = 1000
20 FOR X = 1 TO 10
30   S = S + .08 * S
40 NEXT X
50 PRINT S
```

Note how line 30 takes the old value of S plus 8% of S and adds them to determine the new value of S. Repeating this process 10 times yields the value of the bond.

Draw a chessboard (8x8 squares). On the first square, draw an IOU for one billionth of a dollar (.000000001 dollar). On the second square place twice as much (.000000002 dollar), and on the third place twice as much again (.000000004 dollar). Ask the students which they would prefer - the value of the chessboard once filled in this manner or one hundred dollars. The solution is found from:

```
10 N = 1E - 9
20 FOR I = 1 TO 64
30   S = S + N
40   N = N * 2
50 NEXT I
60 PRINT "ACCUMULATED AMOUNT IS";S
```

Note how line 40 doubles the amount which will be added on the subsequent time around the loop. When the program is run, the result is found to be

```
RUN
ACCUMULATED AMOUNT IS 1.84467441E+10
```

So the chessboard is worth over 18 million dollars!



# Worksheet 3.2

Part I Predict the output of each of the following programs:

- a) 

```
10 A = 5
20 FOR X = 1 TO 3
30   PRINT "A EQUALS";A
40   A = A + 5
50 NEXT X
```
- b) 

```
10 X = 1
20 FOR Y = 1 TO 4
30   PRINT X
40   X = X + 1
50 NEXT Y
```
- c) 

```
10 X = 0
20 Y = 3
30 X = X + 1
40 IF X > Y THEN 70
50 PRINT X;"IS NOT GREATER THAN";Y
60 GOTO 30
70 PRINT X;"IS GREATER THAN";Y
```
- d) 

```
10 X = 0
20 FOR I = 1 TO 5
30   X = X + I
40 NEXT I
50 PRINT X
```
- e) 

```
10 X = 5
20 PRINT X
30 IF X = 12 THEN 70
40 IF X <= 10 THEN 60
50 X = X - 9: GOTO 20
60 X = X + 5: GOTO 20
70 PRINT "YEAH"
```
- f) 

```
10 FOR I = 1 TO 3
20   READ N,M
30   S = S + 2 * M + N
40 NEXT I
50 PRINT S
60 DATA 3,-2,1,5,-6,2
```

Part II Write each of the following programs as concisely as possible:

- a) Write a program to count the number of occurrences of 7 in 100 rolls of a pair of dice.

```
RUN
7 OCCURRED 22 TIMES
```

- b) If your father gave you 1 penny on the first day, 2 on the second, and 4 on the third, and so on, doubling the money given you per day, how much money would you have at the end of 31 days?

# Worksheet 3.2 Solutions

## Part I

a) A EQUALS 5  
A EQUALS 10  
A EQUALS 15

b) 1  
2  
3  
4

c) 1 IS NOT GREATER THAN 3  
2 IS NOT GREATER THAN 3  
3 IS NOT GREATER THAN 3  
4 IS GREATER THAN 3

d) 15

e) 5  
10  
15  
6  
11  
2  
7  
12  
YEAH

f) The student should test each value of I separately:  
When I=1,  $2*M+N = -1$ , so S=-1  
When I=2,  $2*M+N = 11$ , so S=10  
When I=3,  $2*M+N = -2$ , so S=8  
Consequently, the output of the program is:

8

## Part II

a) 10 FOR I = 1 TO 100  
20 IF ( INT ( RND (0) \* 6 + 1) + INT ( RND (0) \* 6 +  
1) ) = 7 THEN N = N + 1  
30 NEXT I  
40 PRINT "7 OCCURRED";N;"TIMES"

b) 10 FOR I = 1 TO 31  
20 J = J + 2 ↑ (I - 1)  
30 NEXT I  
40 PRINT J / 100;"DOLLARS"

RUN  
21474836.5 DOLLARS



# Lesson 3.3

- OBJECTIVES:
- a) to learn the limitations in precision and dynamic range associated with the computer
  - b) to encounter a situation where 'rounding error' is a consideration
  - c) to learn the effects of TIME and POS(0)
  - d) to create both arbitrary and definite length time delays during program execution
  - e) to contrast the use of INPUT and GET
- 

ASSIGNMENT: Read pages 3.4 through 3.6  
Text problems 21,23                      Homework problem 8

---

The material in this lesson presents a detailed analysis of rounding errors and is meant primarily for students who may ultimately perform mathematics and science applications on the computer. For younger students or those who are not mathematically inclined, this lesson might be skipped.

The worksheet for Lesson 3.3 is designed for use as presented below. The students should actually run the programs themselves as the lesson proceeds to experience the problems created by rounding errors.

Program 3.5 in the text demonstrates that the summation technique may provide a slightly erroneous result when it performs a great number of operations on a repeating decimal. As discussed in the text, all numbers are repeating decimals on the computer except those in the set:

1,2,4,8,16,32,64,...

The problem is more extensive than is first evident. The computer prints no more than nine digits, independent of the position of the decimal point within those digits. These digits are called the mantissa. Where needed, the mantissa is modified by an exponent E. There is at least one additional mantissa digit retained but not printed. This can be illustrated using Worksheet 3.3, problem (a). The output begins as follows:

ITERATION	RESULT
1	1.00000001
2	1.00000001
3	1.00000001
4	1.00000001
5	1.00000001
6	1.00000002
7	1.00000002
8	1.00000002
9	1.00000002
10	1.00000002

PRESS 'E' TO END, 'RETURN' TO CONTINUE

The variable X initially contains the value 1.00000001. At each iteration the value .00000001 is added to X. Even though iterations 1 through 5 seem to indicate that no change is occurring in the summed value, the hidden digit(s) must be changing, and this change is revealed at iteration 6. If the calculations are done by hand, it becomes apparent that the PRINT statement rounds numbers to 9 digits (e.g. iterations 1 through 5 are rounded down; iterations 6 through 9 are rounded up; iterations 11-15 are rounded down, etc.)

The hidden digits can cause a further problem exemplified by Worksheet 3.3, problem (b). The output will be as follows:

```
RUN
 11
RESULT IS NOT 11
```

Rounding error has accumulated in the hidden digit(s) so that although the value B is printed as 11, B really contains a number slightly different from 11. Adding the following lines to Worksheet 3.3, problem (b) will round off the hidden digit(s).

```
60 B = INT (B * 1E8 + .5) / 1E8
70 PRINT B
80 IF B = 11 THEN PRINT "NOW RE
   RESULT IS 11"
```

```
RUN
 11
RESULT IS NOT 11
 11
NOW RESULT IS 11
```

Line 60 rounds off the value of B to 7 decimal places. The '1E8' may have to be altered for other problems, depending on the nature of the problem and the severity of the rounding error.

From algebra,

$$(\sqrt{X})^2 = X$$

but the presence of rounding error is evident again in Worksheet 3.3, Problem (c). Part of the output follows. SQR is introduced later in the text, but it is not difficult for students to handle here:

<u>_X</u>	<u>(SQR(X)) ^2</u>	<u>MATCH?</u>
1	1	*** MATCH ***
2	2	*** MATCH ***
3	3	NO MATCH
4	4	*** MATCH ***
5	5.00000001	NO MATCH
6	6.00000001	NO MATCH
7	7.00000001	NO MATCH
8	8	*** MATCH ***
9	9.00000001	NO MATCH
10	10	NO MATCH

PRESS 'E' TO END, 'RETURN' TO CONTINUE

Notice that the algebraic fact  $(\sqrt{X})^2 = X$  is only indicated when X is an integral power of 2. Rounding error exists in all other cases. The rounding error is typically small enough to be ignored by the PRINT statement but is always detected by the conditional statement (IF).

One final note is in order. It was noted earlier that the computer never prints more than nine digits (excluding the exponent). We can say that the mantissa contains nine digits of precision. Of course, this does not mean that 999999999 is the largest number recognized by the computer. Because the mantissa is, in fact, modified by an exponent, numbers considerably larger (or smaller) can be used. Worksheet 3.3, Problem (d) illustrates that there is an ultimate limit to the size of this exponent. Running the program produces the following output:

```

RUN
1      10
100    1000
10000  100000
1000000 10000000
100000000 1000000000
1000000000 1E+09
1E+10   1E+11
1E+12   1E+13
1E+14   1E+15
1E+16   1E+17
1E+18   1E+19
1E+20   1E+21
1E+22   1E+23
1E+24   1E+25
1E+26   1E+27
1E+28   1E+29
1E+30   1E+31
1E+32   1E+33
1E+34   1E+35
1E+36   1E+37
1E+38
?OVERFLOW ERROR IN 34

```

Similarly, the lower limit is found by changing the operation in lines 30 and 40 to division. One difference in the output is that when the exponent becomes too small, no error results. Instead, 0 is assigned to the variable in question.

Using the results of the above program, we can say that the 'dynamic range' of the computer is about  $10^{(-38)}$  to  $10^{(38)}$ .

Many times you may find that the output you so carefully designed is "scrolled up" out of sight by subsequent PRINT or INPUT statements. A "delay" loop can postpone the inevitable from happening. A delay loop can take on two forms on the Commodore 64. The simplest form is that of a FOR...NEXT loop which uses time. In other words, no operations are performed inside the loop. Such a loop would take the form:

```
100 FOR X = 1 TO 500: NEXT X
```

The values of the loop variable are totally arbitrary and are only dependent upon the programmer and the amount of time he/she wishes to stall processing.

The second type of delay loop can be considered a definite length delay loop. It makes use of the built-in TIME variable. The TIME variable is set to the number of "jiffies" elapsed since the most recent power-up. As explained in the text, a "jiffy" is equal to 1/60 of a second. Using this fact, we can construct the following delay loop:

```
100 REM 10 SECOND DELAY LOOP
110 T = TIME
120 IF INT((TIME - T) / 60) <> 10 THEN 120
130     .
140     .
150     .
```



# Worksheet 3.3

Part I Examine the output of each of the following programs by typing them into the computer and running them:

- a)
- ```
10 X = 1.00000001
20 T = .000000001
30 PRINT "ITERATION    RESULT"
40 PRINT "-----    ----"
50 X = X + T
60 I = I + 1
70 PRINT I; TAB( 12);X
80 IF I / 20 = INT (I / 20) THEN 100
90 GOTO 20
100 PRINT : PRINT "PRESS 'E' TO END, 'RETURN' TO CONTINUE";
110 GET Z$: IF Z$ = "" THEN 110
120 IF Z$="E" THEN END
130 GOTO 30
```
- b)
- ```
10 FOR X = 1 TO 33
20   B = B + 1 / 3
30 NEXT X
40 PRINT B
50 IF B < > 11 THEN PRINT "RESULT IS NOT 11"
```
- c)
- ```
10 PRINT " X          (SQR(X))↑2          MATCH?"
20 PRINT "----          -----          ----"
30 X = X + 1
40 Y = SQR (X)
50 Z = Y ↑ 2
60 PRINT X; TAB( 8);Z;
70 M$ = "NO MATCH"
80 IF Z = X THEN M$ = "*** MATCH ***"
90 PRINT TAB(23);M$
100 IF X / 10 = INT (X / 10) THEN 120
110 GOTO 20
120 PRINT : PRINT "PRESS `E` TO END, 'RETURN' TO CONTINUE";
130 GET Z$: IF Z$ = "" THEN 130
140 IF Z$ <> "E" THEN PRINT : GOTO 10
150 END
```
- d)
- ```
10 X = .1
20 FOR I = 1 TO 20
30   X = X * 10
40   PRINT X,
50   X = X * 10
60   PRINT X
70 NEXT I
```

# Worksheet 3.3 Solutions

NOTE: The solutions for Worksheet 3.3 have been supplied as an integral portion of this lesson's text.





- 6) Your father gives you a penny as a gift on your first birthday. He promises to double the amount of the gift each year until your 21st birthday. Have the computer print a table showing the amount you receive on each birthday and the total you will have received as of each year.
- 7) A well-known 'infinite series' is

$$\frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \frac{1}{16} + \dots$$

Write a program to add repeatedly the next term in the series and to print the sum thus far after each addition. Your program should be an infinite loop, but it will be clear that after a while the sum of the series is 1.

- 8) Write a program which will find integer solutions of the equation

$$x^2 = 5 * x$$

for  $-10 \leq x \leq 10$ . Run the program and discuss the output. Is anything different from what you might expect?

# Chapter 3 Supplementary Problem Solutions

- 1) 

```
10 FOR I = 1 TO 6
20   PRINT TAB( I); "*"
30 NEXT I
40 FOR I = 5 TO 1 STEP -1
50   PRINT TAB( I); "*"
60 NEXT I
```
- 2) 

```
10 INPUT "NUMERATOR";N
20 INPUT "DENOMINATOR";D
30 PRINT "REMAINDER IS";N / D - INT (N / D)
```
- 3) 

```
10 INPUT "HOW MANY SIDES";S
20 INPUT "HOW MANY THROWS";T
30 N = 0
40 FOR I = 1 TO T
50   N = N + INT (RND (1) * S + 1)
60 NEXT I
70 PRINT "THE SUM IS";N
```
- 4) 

```
10 INPUT "YEAR";Y
20 IF Y / 4 = INT (Y / 4) THEN PRINT "IT IS A LEAP YEAR": END
30 PRINT "IT IS NOT A LEAP YEAR"
```
- 5) 

```
10 M = 1
20 INPUT "CURRENT YEAR";N
30 PRINT "YEAR", "BALANCE"
40 FOR Y = 1955 TO N
50   M = M * 1.05
60   PRINT Y, M
70 NEXT Y
```
- 6) 

```
10 PRINT "RECEIVED", "TOTAL"
20 FOR I = 1 TO 21
30   S = S + 2 ↑ (I - 1)
40   PRINT 2 ↑ (I - 1), S
50 NEXT I
```
- 7) 

```
10 X = .5
20 Y = .5
30 Y = Y / 2
40 X = X + Y
50 PRINT X
60 GOTO 30
```
- 8) 

```
10 FOR I = -10 TO 10
20   IF (I ↑ 2) = (5 * I) THEN PRINT I; "IS A SOLUTION"
30 NEXT I
```



# Chapter 3 Test

Part I Determine the exact output of each of the following programs:

- a) 10 FOR I = 1 TO 3  
20 READ X  
30 PRINT INT (X)  
40 NEXT I  
50 DATA -1.1,.1,1.5,101
- b) 10 FOR J = 1 TO 5  
20 S = S + J  
30 NEXT J  
40 PRINT S
- c) 10 X = 1  
20 S = S + X  
30 X = X \* 2  
40 IF S < 12 THEN 20  
50 PRINT X,S
- d) 10 PRINT "12345"  
20 PRINT POS(0)  
30 PRINT "ABCDE";  
40 PRINT POS(0)

Part II Circle the programming error(s) within each program. Explain what must be done to correct the error. If no error exists, state so.

- a) 10 REM COUNT BY 10'S STARTING WITH 10  
20 S = 1  
30 S = S \* 10  
40 PRINT S  
50 GOTO 30
- b) 10 PRINT "TYPE 'NO' TO END PROGRAM";  
20 GET X\$  
30 IF X\$ = "NO" THEN END  
40 GOTO 10
- c) 10 REM PRINT SUM OF 1000 RANDOM NUMBERS  
20 REM BETWEEN -10 AND 10, INCLUSIVE  
30 FOR I = 1 TO 1000  
40 S = S + INT ( RND (1) \* 20) - 10)  
50 NEXT I
- d) 10 REM DETERMINE "ODD" OR "EVEN"  
20 INPUT X  
30 IF X / 2 < > INT (X) THEN 50  
40 PRINT "EVEN": GOTO 20  
50 PRINT "ODD": GOTO 20

Part III Write each of the following programs as concisely as possible:

- a) Write a program to find the sum of the first 100 terms in the series

$$1X + 2X + 3X + 4X + \dots$$

WHERE  $X = .4$

- b) Write a program to simulate the rolling of two dice that will produce an output similar to the RUN shown below. If "doubles" are rolled (both die give the same value), then the player gets a kewpie doll; otherwise, nothing is won and the program ends.

RUN

DIE 1 IS A 5  
DIE 2 IS A 5

YOU'VE WON A KEWPIE DOLL!!

DIE 1 IS A 3  
DIE 2 IS A 2

YOU'VE WON NOTHING!!

# Chapter 3 Test Solutions

The credit for each problem is given in brackets [ ].

Part I [ 6@ = 24 ]

- |    |                     |    |                              |    |
|----|---------------------|----|------------------------------|----|
| a) | RUN<br>-2<br>0<br>1 | c) | RUN<br>16                    | 15 |
| b) | RUN<br>15           | d) | RUN<br>12345<br>0<br>ABCDE 5 |    |

Part II [ 6@ = 24 ]

- a) In order to count by 10's, change line 20 to S=0 and line 30 to S = S + 10.
- b) In line 20 the INPUT command should be used instead of the GET command. Remember the GET command only accepts a single character string variable.
- c) In line 40 the statement (RND(0) \* 20) should be (RND(0) \* 21).
- d) In line 30 the statement INT (X) should be INT (X/2).

Part III

- a) 10 FOR I = 1 TO 1000 [ 22 ]  
20 X = X + (I \* .4 ↑ (I - 1))  
30 NEXT I  
40 PRINT X
- b) 10 D1 = INT(RND(0) \* 6) + 1 [ 30 ]  
20 D2 = INT(RND(0) \* 6) + 1  
30 PRINT "DIE 1 IS A";D1  
40 PRINT "DIE 2 IS A";D2: PRINT  
50 IF D1 < > D2 THEN 80  
60 PRINT "YOU'VE WON A KEWPIE DOLL":PRINT  
70 GOTO 10  
80 PRINT "YOU'VE WON NOTHING"  
90 END





# Lesson 4.1

OBJECTIVE: a) to comprehend the order of statement execution for nested loop programs

---

ASSIGNMENT: Read pages 4.1 and 4.2 Do review problem 1  
Text problems 1,7a,7b,9,13 Homework problems 2,8

---

In Chapter 2 the concept of entering and exiting loops was introduced. Again, proper indenting with FOR...NEXT loops should always be followed. Consider the following nested loop program:

```
10 REM NESTED LOOP DEMO
20 FOR I = 1 TO 3
30 PRINT "I IS NOW";I
40 FOR J = 1 TO 3
50 PRINT " J IS NOW";J
60 NEXT J
70 NEXT I
80 PRINT "BOTH LOOPS COMPLETED"
```

Anytime the 'J' loop is entered, you cannot exit the loop until it is completed. So, line 70 will not be executed until the variable J has taken on the values 1,2 and then 3. After line 70 has executed for the first time (I takes on the value 2), line 40 is executed again. We have entered the 'J' loop a second time. The 'J' loops start over.

Transparency Master 4.1 may be of assistance during your discussion. The output is as follows:

```
RUN
I IS NOW 1
  J IS NOW 1
  J IS NOW 2
  J IS NOW 3
I IS NOW 2
  J IS NOW 1
  J IS NOW 2
  J IS NOW 3
I IS NOW 3
  J IS NOW 1
  J IS NOW 2
  J IS NOW 3
BOTH LOOPS COMPLETED
```

Try inserting inconspicuous loops at line 35 and 55 so that the variable value changes can be discussed as they occur.

```
35 FOR A = 1 TO 2000 : NEXT A
55 FOR B = 1 TO 1000 : NEXT B
```

Be sure to point out that whereas single loops could be used to solve equations with one unknown by trial and error, nested loops can be used to solve equations with multiple unknowns. Program 4.2 illustrates this technique.

# Transparency 4.1

```
10 REM NESTED LOOP DEMO
```

```
20 FOR I = 1 TO 3  
30   PRINT "I IS NOW";I
```

```
40   FOR J = 1 TO 3
```

```
50     PRINT "   J IS NOW";J
```

```
60   NEXT J
```

```
70 NEXT I
```

```
80 PRINT "BOTH LOOPS COMPLETED"
```



# Worksheet 4.1

Part I Determine the exact output of each of the following programs:

a) 10 FOR I = 1 TO 3  
20 FOR J = 2 TO 4  
30 PRINT J  
40 NEXT J  
50 NEXT I

b) 10 FOR I = 1 TO 3  
20 FOR J = 2 TO 4  
30 PRINT I  
40 NEXT J  
50 NEXT I

c) 10 FOR A = 1 TO 3  
20 FOR B = 3 TO 1 STEP -1  
30 PRINT A + B  
40 NEXT B  
50 NEXT A

d) 10 FOR F = -3 TO 4  
20 FOR K = 6 TO 8  
30 S = S + 1  
40 NEXT K  
50 NEXT F  
60 PRINT S

e) 10 FOR M = 1 TO 2  
20 FOR N = 1 TO 2  
30 PRINT N \* M  
40 NEXT M  
50 NEXT N

f) 10 FOR P = 1 TO 3  
20 FOR Q = 1 TO 3  
30 PRINT TAB(Q);P  
40 NEXT Q  
50 NEXT P

g) 10 FOR P = 1 TO 3  
20 FOR Q = 1 TO 3  
30 PRINT TAB(P);Q  
40 NEXT Q  
50 NEXT P

h) 10 FOR I = 2 TO 4  
20 FOR J = 0 TO 1  
30 PRINT J;I  
40 NEXT J  
50 NEXT I

i) 10 FOR X = 3 TO 12 STEP 3  
20 FOR Y = 12 TO 3 STEP -3  
30 IF X > Y THEN PRINT X  
40 NEXT Y  
50 NEXT X

j) 10 K = - 1  
20 FOR I = - 1 TO 1  
30 FOR J = 1 TO -1 STEP -1  
40 PRINT I \* J \* K  
50 NEXT J  
60 K = K + 1  
70 NEXT I

k) 10 FOR I = 1 TO 2  
20 FOR J = 1 TO 2  
30 FOR K = 1 TO 2  
40 PRINT I;"/";J;"/";K  
50 NEXT K  
60 NEXT J  
70 NEXT I

l) 10 FOR I = 1 TO 3  
20 PRINT J  
30 FOR J = 2 TO 4  
40 NEXT J  
50 NEXT I

# Worksheet 4.1 Solutions

## Part I

a)	RUN	b)	RUN	c)	RUN	d)	RUN
	2		1		4		24
	3		1		3		
	4		1		2		
	2		2		5		
	3		2		4		
	4		2		3		
	2		3		6		
	3		3		5		
	4		3		4		

e) RUN  
1  
2

?NEXT WITHOUT FOR ERROR IN 50

f)	RUN	g)	RUN	h)	RUN	i)	RUN
	1		1		0 2		6
	1		2		1 2		9
	1		3		0 3		9
	2		1		1 3		12
	2		2		0 4		12
	2		3		1 4		12
	3		1				
	3		2				
	3		3				

j)	RUN	k)	RUN	l)	RUN
	1		1 / 1 / 1		0
	0		1 / 1 / 2		5
	-1		1 / 2 / 1		5
	0		1 / 2 / 2		
	0		2 / 1 / 1		
	0		2 / 1 / 2		
	1		2 / 2 / 1		
	0		2 / 2 / 2		
	-1				

### Notes:

- d) Line 30 is a simple counter. The F loop will execute 8 times, and for each F iteration the K loop must execute 3 times. So,  $8 * 3 = 24$ .
- e) The actual output is rather unexpected, but students should notice that the loops are 'crossed'.
- l) Two points are of interest here. First, like any other numeric variable, a loop variable is set to zero at the beginning of program execution. Second, the computer increments the loop variable, then checks to see if its value is beyond the loop variable limit.

## Lesson 4.2

- OBJECTIVES: a) to understand why subscripted variables are important
- b) to use subscripted variables in programming situations
- 

ASSIGNMENT: Read pages 4.3 through 4.9 Do review problems 2-4  
Text problems 3,7c,11,19,23 Homework problems  
10,12,20,22

---

The text discusses the need for subscripted variables by showing the difficulty of avoiding repetitions in a list of random numbers. The text is clear and thorough on this matter so an alternate approach is presented here:

Suppose nine art experts are attending an auction. A priceless painting is to be sold by a secret bid, and the names and the bids are to be stored in a computer until the bidding is completed. The novice may write a program such as the following:

```
10 INPUT "ENTER NAME";N1$
12 INPUT "ENTER BID";B1
15 FOR J = 1 TO 25: PRINT: NEXT J
20 INPUT "ENTER NAME";N2$
22 INPUT "ENTER BID";B2
15 FOR J = 1 TO 25: PRINT: NEXT J
.
.
90 INPUT "ENTER NAME";N9$
92 INPUT "BID";B9
94 FOR J = 1 TO 25: PRINT: NEXT J
98 REM
100 INPUT "PUSH 'RETURN' TO REVEAL BIDS";Z$
102 REM
200 PRINT "NAME","BID"
202 PRINT "----","----"
210 PRINT N1$,B1
220 PRINT N2$,B2
.
.
290 PRINT N9$,B9
```

Each art expert will walk up to the computer and enter his or her bid. Of course, the screen clears immediately after the bid entry.

The student can see that this requires 18 different variable names and a lot of typing. If there were more than nine bidders, the simple sequence of variable names would be lost because N1\$ and N10\$ are not two different variables. Even for nine bidders it would be difficult to revise the program so that the high bidder would be determined after all the bids are in. And what if there were 50 bidders? Or 1000 bidders? No one would care to do all the typing!

Instead of simple variables, subscripted variables will be used.  
So,

```
N1$,N2$,N3$,...,N9$  
B1,B2,B3,....,B9
```

are replaced by

```
N$(1),N$(2),N$(3),...,N$(9)  
B(1),B(2),B(3),....,B(9)
```

The significance of subscripted variables is that the subscript can be replaced by a variable:

```
N$(I)  
B(I)
```

Can your students suggest a simple means for changing the value from 1 to 9 so that the names and bids can be entered?

A FOR...NEXT loop is used to alter the subscript, and the program becomes:

```
10 FOR I = 1 TO 9  
20   INPUT "ENTER NAME";N$(I)  
30   INPUT "ENTER BID";B(I)  
40   FOR J = 1 TO 25: PRINT: NEXT J  
50 NEXT I  
100 INPUT "PUSH 'ENTER' TO REVEAL BIDS";Z$  
110 FOR J = 1 TO 25: PRINT: NEXT J  
200 PRINT "NAME","BID"  
210 PRINT "----","----"  
220 FOR I = 1 TO 9  
230   PRINT N$(I),B(I)  
240 NEXT I
```

Your students would enjoy actually performing the secret bidding as a means of testing the program. To make the situation even more instructive, first add the lines:

```
12 PRINT "THE SUBSCRIPT IS NOW";I  
14 PRINT "YOU ARE ABOUT TO ENTER N$( ";I;" )"  
25 PRINT "YOU ARE ABOUT TO ENTER B( ";I;" )"
```

Transparency 4.2 will be a valuable aid for the discussion above. Fill in the boxes with the input data as the program is run to reinforce the box analogy.

Note that there is a box with subscript zero in any array (N\$(0),B(0)) which can be used. Negative subscripts are not permissible. If for some reason a subscript is not an integer, only the integer portion is considered during execution.

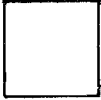


# Transparency 4.2

N\$(0)



N\$(1)



N\$(2)



N\$(3)



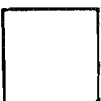
N\$(4)



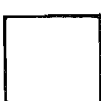
N\$(5)



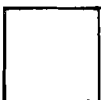
N\$(6)



N\$(7)



N\$(8)



N\$(9)



B(0)



B(1)



B(2)



B(3)



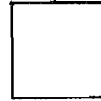
B(4)



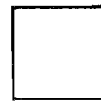
B(5)



B(6)



B(7)



B(8)



B(9)





# Worksheet 4.2

Part I Determine the exact output of each of the following programs:

- a) 

```
10 FOR I = 1 TO 10
20   X(I) = I * 2 - 5
30 NEXT I
40 FOR I = 1 TO 10
50   PRINT X(I)
60 NEXT I
```
- b) 

```
10 FOR J = 1 TO 10 STEP 2
20   A(J) = J
30 NEXT J
40 FOR K = 5 TO 1 STEP - 1
50   PRINT A(K)
60 NEXT K
```
- c) 

```
10 FOR X = 1 TO 5
20   READ A$(X)
30 NEXT X
40 FOR J = 2 TO 4
50   PRINT A$(J)
60 NEXT J
70 DATA APE,BAT,CAT,DOG,EEL
```
- d) 

```
10 FOR X = 1 TO 5
20   READ W$(X)
30 NEXT X
40 FOR Y = 5 TO 1 STEP -1
50   PRINT W$(Y)
60 NEXT Y
70 DATA ONE,TWO,THREE,FOUR,
   FIVE
```
- e) 

```
10 FOR I = 1 TO 3
20   A$(1) = "HA!"
30 NEXT I
40 PRINT A$(3)
```
- f) 

```
10 FOR J = 1 TO 7
20   READ X(J)
30 NEXT J
40 FOR K = 1 TO 10 STEP 2
50   T = T + X(K)
60 NEXT K
70 PRINT T
80 DATA 2,3,5,17,1,6,10,
   -50,4,8
```
- g) 

```
10 FOR I = 1 TO 5
20   C(I+1) = I-1
30 NEXT I
40 FOR J = 1 TO 7
50   PRINT C(J)
60 NEXT J
```
- h) 

```
10 G(1) = 2
20 FOR F = 2 TO 4
30   G(F) = G(F-1)↑2
40 NEXT F
50 FOR F = 1 TO 4
60   PRINT G(F)
70 NEXT F
```

# Worksheet 4.2 Solutions

## Part I

a)	RUN	b)	RUN	c)	RUN	d)	RUN
	-3		5		BAT		FIVE
	-1		0		CAT		FOUR
	1		3		DOG		THREE
	3		0				TWO
	5		1				ONE
	7						
	9						
	11						
	13						
	15						
e)	RUN	f)	RUN	g)	RUN	h)	RUN
			18		0		2
	READY.				0		4
			READY.		1		16
					2		256
					3		
					4		
					0		

### Notes:

- b) The zeroes are present because all numeric variables are set to zero at the beginning of program execution. A(5), A(3) and A(1) are not altered by the FOR...NEXT loop from lines 10 to 30.
- e) No output. Many students will be unobservant and interpret line 20 as A\$(I)="HA!".
- g) Students have seen that the subscript can be a constant numeric value or a variable. Here the subscript is an algebraic expression.

# Lesson 4.3

OBJECTIVE: a) to extend understanding of subscripted variables to include double subscripts

---

ASSIGNMENT: Read pages 4.10 through 4.13 Do review problems 5 - 6  
Text problems 5,17,21 Homework problems 6,14,16,18,24

---

The concept of doubly subscripted variables can be difficult for students, but it is valuable especially for string data that can be represented in a "real world" situation by rows and columns. For example, seats on an airliner, or post office boxes usually are set up in rows and columns.

Review the diagram on page 4.11 in the text. Make sure that your students understand that the rows run horizontally and the columns vertically and that the subscript gives the row and then the column.

A(row,column)

---

It is advisable to spend time on Program 4.10 stepping through it line by line since this program reviews most of the programming concepts covered to this point in the text. Emphasize the necessity of a DIM statement for all doubly subscripted variables; without it the computer does not know how the rows and the columns relate.

The FOR...NEXT loop between lines 30 and 90 is the heart of Program 4.10. Explain to your students that after line 40 reads a name from the DATA statement, lines 50 and 60 randomly pick a row and column in which the name is to be stored. Line 70 checks if that position has been previously chosen. If it is empty, N\$(R,C) will contain a blank space which the computer represents as " "; if it has any other string information stored in N\$(R,C), the program returns to line 50 to pick another random row and column. As the program operates, the computer is going to find it increasingly difficult to locate an empty seat randomly. To find out how many attempts the computer made to fill the 14 seats, add a counter at line 65

```
65 Z = Z + 1
```

and print the result at line 95

```
95 PRINT Z;"ATTEMPTS WERE MADE TO FILL THE 14 SEATS"
```

---

To further assist students in seeing how Program 4.10 works, add the following lines and have the students fill in the appropriate boxes in the doubly subscripted array as it is actually run:

```
85 PRINT "ROW";R,"SEAT";C
86 PRINT "HAS BEEN ASSIGNED TO";N$(R,C)
88 INPUT "HIT RETURN TO CONTINUE";D$
```

Notice how line 88 will stop the program to give the user time to read the information and will continue only after a RETURN is typed. The string variable DUMMY\$ is used only to complete the INPUT statement; any unused string variable can be used.

A complete listing of Program 4.10 and the boxes needed to fill the program run are given in Transparency Master 4.3.

# Transparency 4.3

```
10 DIM N$(5,3)
20 REM READ 14 NAMES RANDOMLY INTO ARRAY
30 FOR X = 1 TO 14
40   READ A$
50   R = INT (5 * RND(0) + 1)
60   C = INT (3 * RND(0) + 1)
65   Z = Z + 1
70   IF N$(R,C) <> "" THEN 50
80   N$(R,C) = A$
85   PRINT"ROW";R,"SEAT";C
96   PRINT"HAS BEEN ASSIGNED TO ";N$(R,C)
88   INPUT "HIT RETURN TO CONTINUE";D$
90 NEXT X
95 PRINT Z;"ATTEMPTS WERE MADE TO FILL 14 SEATS"
100 REM PRINT SEATING CHARTS
110 FOR R1 = 1 TO 5
120   FOR C1 = 1 TO 3
130     IF N$(R1,C1) = "" THEN PRINT "EMPTY",: GOTO 150
140     PRINTN$(R1,C1),
150   NEXT C1
160 NEXT R1
170 REM
180 DATA ANNE,DON,SHERRY,MAGGIE,TED,LIZ,ROB
190 DATA MARY,DAVID,MARK,KEVIN,SUSAN,WENDELL,CINDY
```

	1	2	3
1	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
2	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
3	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
4	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
5	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>





# Worksheet 4.3

Part I The boxes at the right represent double subscripted arrays. For each program below place the indicated value in the appropriate boxes:

a) 10 N\$(2,2) = "A"  
 20 N\$(1,3) = "B"  
 30 N\$(3,1) = "C"  
 40 N\$(4,2) = "D"

		array N\$				
		column				
		1	2	3	4	5
r o w	1					
	2					
	3					
	4					
	5					

b) 10 FOR I = 1 TO 5  
 20 FOR J = 1 TO 5  
 30 X(I,J) = I \* J  
 40 NEXT J  
 50 NEXT I

		array X				
		column				
		1	2	3	4	5
r o w	1					
	2					
	3					
	4					
	5					

Part II This program should put a numeric value entered at the keyboard into a random position within a two dimensional array. Identify and correct any errors which exist.

a) 10 DIM Y(12,12)  
 20 I = INT ( RND(0) \* 13) + 1  
 30 J = INT ( RND(0) \* 13) + 1  
 40 INPUT W  
 50 Y = W(I,J)

Part III Determine the exact output of each of the following programs:

a) 10 DIM X\$(3,3)  
 15 FOR R = 1 TO 3  
 20 FOR C = 1 TO 3  
 25 READ X\$(R,C)  
 30 NEXT C  
 35 NEXT R  
 40 REM  
 45 FOR C = 1 TO 3  
 50 FOR R = 1 TO 3  
 55 PRINT X\$(R,C);  
 60 NEXT R  
 65 PRINT  
 70 NEXT C  
 75 DATA C,A,T,O,N,E,P,T,A

b) 10 FOR I = 1 TO 3  
 20 FOR J = 1 TO 7  
 30 READ X(I,J)  
 40 NEXT J  
 50 NEXT I  
 60 PRINT X(2,5)  
 70 PRINT X(3,1)  
 80 DATA 3,7,5,6,9,1  
 81 DATA 2,4,11,24,53,31  
 82 DATA 92,93,98,47,42  
 83 DATA 49,16,19,107

# Worksheet 4.3 Solutions

## Part I

a)

		B		
	A			
C				
	D			

b)

1	2	3	4	5
2	4	6	8	10
3	6	9	12	15
4	8	12	16	20
5	10	15	20	25

## Part II

Lines 20 and 30: The list of random numbers includes 13, which is beyond the range established by the DIM statement.

Line 50: Should be  $Y(I,J) = W$

## Part III

a) RUN  
COP  
ANT  
TEA

b) RUN  
31  
98

# Chapter 4 Supplementary Problems

- 1) A computer terminal is installed in a sports arena so that the basketball coaches can keep track of the shooting percentages. We will assume that there are eight players on the squad, with jerseys numbered from one to eight.

For each player you will need to keep track of the number of shots attempted and the number of shots that were successful. At the end of the game or practice the coach will enter a zero as shown in the example, and the computer will print out a table of information.

To calculate the percentage of successful shots, calculate

$$\frac{\text{successful shots}}{\text{shots attempted}} \times 100$$

Watch out for 'divide by zero' error. It will occur if a player takes no shots all game.

```
RUN
INPUT PLAYER NUMBER? 3
DID HE MAKE THE SHOT? Y
INPUT PLAYER NUMBER? 1
DID HE MAKE THE SHOT? Y
INPUT PLAYER NUMBER? 3
DID HE MAKE THE SHOT? N
INPUT PLAYER NUMBER? 5
DID HE MAKE THE SHOT? N
INPUT PLAYER NUMBER? 6
DID HE MAKE THE SHOT? Y
INPUT PLAYER NUMBER? 0
```

PLAYER #	PERCENTAGE
1	100
2	0
3	50
4	0
5	0
6	100
7	0
8	0

- 2) Ignoring leap years, one may assume that babies have a 1-in-365 chance to be born on a given day of the year. Certainly, in a crowd of 366 people, at least two people will have the same birthdate. What do you suppose the chances are that two people will have the same birthdate in a crowd of 35 people? You'll be surprised.
- a) Taking January 1 as day 1 and December 31 as day 365, write a program which will randomly select days of the year until a repeat occurs. The computer should then print out the number of 'people in the crowd'.

```
RUN
THERE ARE 22 PEOPLE IN THE CROWD
```

- b) Have the computer repeat the processes 50 times and print out the average 'number of people'.

```
RUN  
THE AVERAGE IS 26.54
```

- 3) Establish a 5x5 array of string boxes. The array will serve as a game board and may be visualized as follows:


Have the computer pick a random row number (1 to 5), and a random column number (1 to 5). Hide the word "DRAGON" in the single box determined by the row and column numbers drawn. Start the user with 10 points and ask the user to pick a row and column. Check the box indicated to see if it contains the word "DRAGON". If it does, print the user's score and end the game. If the box chosen does not contain the word "DRAGON", subtract one from the user's score and place the word "DRAGON" in the box chosen. The player picks another box to continue the game. As you can see, when the player picks the same box twice, or the box in which "DRAGON" was originally placed, the game ends.

- 4) Find all solutions of the equality  $2X = Y(X-1)$  for X between 1 and 20 (integers, inclusive), and Y between -10 and 10 (integers, inclusive). Use a "looping solution".

# Chapter 4 Supplementary Problem Solutions

```
1) 10 DIM A(10,2)
    20 INPUT "INPUT PLAYER NUMBER";A
    30 IF A = 0 THEN 100
    40 INPUT "DID HE MAKE THE SHOT";A$
    50 IF A$ < > "Y" AND A$ < > "N" THEN 40
    60 IF A$ = "Y" THEN A(A,2) = A(A,2) + 1
    70 A(A,1) = A(A,1) + 1
    80 GOTO 20
    100 PRINT : PRINT "PLAYER #","PERCENTAGE"
    110 FOR I = 1 TO 8
    120   IF A(I,1) = 0 THEN PRINT I,0: GOTO 140
    130   PRINT I,A(I,2) / A(I,1) * 100
    140 NEXT I

2a) 10 DIM A(365)
    20 I = 1:A(1) = INT ( RND(0) * 365 + 1)
    30 A = INT (RND (0) * 365 + 1)
    40 FOR J = 1 TO I
    50   IF A = A(J) THEN 100
    60 NEXT J
    70 A(J + 1) = A:I = I + 1
    80 GOTO 30
    100 PRINT "THERE ARE";I + 1;"PEOPLE IN THE CROWD"

b) 10 DIM A(365)
    15 FOR K = 1 TO 50
    20   I = 1:A(1) = INT ( RND(0) * 365 + 1)
    30   A = INT (RND (0) * 365 + 1)
    40   FOR J = 1 TO I
    50     IF A = A(J) THEN 100
    60   NEXT J
    70   A(J + 1) = A:I = I + 1
    80   GOTO 30
    100  N = N + I + 1
    110 NEXT K
    120 PRINT "THE AVERAGE IS";N / 50

3) 10 DIM A$(5,5): S = 10
    20 A$( INT ( RND (0) * 5 + 1), INT ( RND (0) * 5 + 1)) = "DRAGON"
    30 INPUT "(ROW,COLUMN)";X,Y
    40 IF A$(X,Y) = "" THEN A$(X,Y) = "DRAGON":S = S - 1: GOTO 30
    50 PRINT "YOUR SCORE IS ";S

4) 10 FOR X = 1 TO 20
    20   FOR Y = -10 TO 10
    30     IF 2 * X = Y * (X - 1) THEN PRINT "X=";X;" Y=";Y;" IS
        A SOLUTION"
    40   NEXT Y
    50 NEXT X

RUN
X= 2 Y= 4 IS A SOLUTION
X= 3 Y= 3 IS A SOLUTION
```



# Chapter 4 Test

Part I Determine the exact output of each of the following programs:

- a) 

```
10 FOR X = 1 TO 3
20   FOR Y = 1 TO 3
30     PRINT X;
40   NEXT Y
50 NEXT X
```
- b) 

```
10 FOR X = 1 TO 4
20   FOR Y = 1 TO 4
30     PRINT X + Y
40   NEXT Y
50 NEXT X
```
- c) 

```
10 FOR P = 1 TO 15
20   FOR Q = 0 TO 5
30     S = S + 1
40   NEXT Q
50 NEXT P
60 PRINT S
```
- d) 

```
10 FOR R = 1 TO 3
20   FOR C = 1 TO 10
30     X(R,C) = R - C
40   NEXT C
50 NEXT R
60 PRINT X(7,2)
70 PRINT X(3,5)
```
- e) 

```
10 FOR J = 1 TO 4
20   READ V,I
30   X(I) = V
40   Y(I) = J
50   Z(J) = I
60 NEXT J
70 FOR J = 1 TO 4
80   PRINT X(J),Y(J),Z(J)
90 NEXT J
100 DATA 3,1,2,2,2,3,4,2
```

Part II Circle the programming error(s) within each program given. Explain what must be done in order to correct the error. If no errors exist, state so.

- a) 

```
10 FOR X = 2 TO 3
20   FOR Y = 4 TO 6
30     PRINT X;Y
40   NEXT X
50 NEXT Y
```
- b) 

```
10 FOR I = 1 TO 20
20   X(I) = INT ( RND
      (0) * 5) + 1
30 NEXT I
```
- c) 

```
10 FOR I = 1 TO 5
20   READ X
30   N(X) = 17
40 NEXT I
50 DATA 2,1,0,-1,-2
```
- d) 

```
10 FOR I = 1 TO 5
20   READ X
30   X↑2 = N(I)
40 NEXT I
50 DATA 2,1,0,-1,-2
```
- e) 

```
10 REM ENTER FIVE WORDS
   AND PRINT THEM OUT IN
   REVERSE ORDER
20 FOR I = 1 TO 5
30   INPUT W$
40 NEXT I
50 FOR I = 5 TO 1 STEP - 1
60   PRINT W$
70 NEXT I
```

Part III Write each of the following programs as concisely as possible:

- a) Find the integers X and Y for the equation below such that both values lie between 0 and 10
- $$2X - Y = -2 \text{ and } -3X + 2Y = 5$$
- b) Write a program to place 20 random integers between 1 and 100, inclusive, in a singly subscripted array called X and then print them out. Let DIM X(20) be the first line in your program.

# Chapter 4 Test Solutions

The credit for each program is given in brackets [ ].

Part I [ 5@ = 25 ]

- a) RUN  
1 1 1 2 2 2 3 3 3
- b) RUN  
2  
3  
4  
5  
3  
4  
5  
6  
4  
5  
6  
7  
5  
6  
7  
8
- c) RUN  
90
- d) RUN  
0  
-2
- e) RUN  
3 1 1  
4 4 2  
2 3 3  
0 0 2

Part II [ 5@ = 25 ]

- a) Lines 40 and 50 are interchanged; therefore a ?NEXT WITHOUT FOR ERROR IN 50 will occur.
- b) Array X() needs to be dimensioned since the subscript goes above 10.
- c) Negative subscripts are illegal.
- d) Expressions on the left side of an assignment statement are illegal; therefore, line 30 should read  
30 N(I) = X ^ 2
- e) Subscripts from line 30 and 60 are missing; they should read:  
30 INPUT W\$(I)  
60 PRINT W\$(I)







# Lesson 5.1

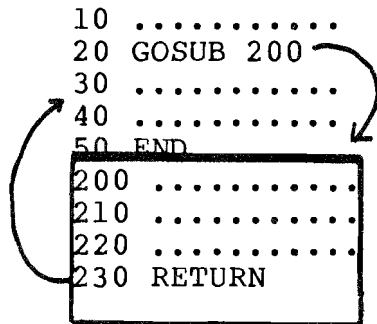
- OBJECTIVES: a) to contrast the difference between branching using GOTO and branching using GOSUB
- b) to be able to incorporate ON...GOTO and ON...GOSUB statements into programs

---

ASSIGNMENT: Read pages 5.1 through 5.4      Do review problems 1-3  
Text problems 1,3                              Homework problems 2,4

---

When GOSUB...RETURN is introduced in the text, the skeleton program shown below is introduced. The added arrows and 'box' will help your students comprehend the structure.



However, the real value of GOSUB...RETURN is that a subroutine can be called from any line in a program, and the computer will always return after executing the subroutine to continue from the line following the GOSUB statement. Such a program cannot be written with only GOTOs. For example, consider the following skeleton program:

```
10 .....
20 GOSUB 200
30 .....
40 GOSUB 200
50 .....
60 END
200 .....
210 .....
220 .....
230 RETURN
```

The order of execution of this program is 10,20,200,210,220,230,30,40,200,210,220,230,50,60. Note how line 60 keeps the program from continuing to execute.

---

The variable in the ON <variable> GOTO and the ON <variable> GOSUB statements may only take on positive integer values from 1 to 255. Integer values greater than 255 or less than 0 create an "ILLEGAL QUANTITY ERROR".



# Worksheet 5.1

Part I Determine the exact output of the following programs:

- a) 

```
10 PRINT "A";
20 GOSUB 100
30 PRINT "B";
40 GOSUB 200
50 PRINT "C";
60 END
100 PRINT "D";
110 RETURN
200 PRINT "E";
210 RETURN
```
- b) 

```
10 PRINT "A";
20 X = X + 1
30 ON X GOTO 40,70,150
40 PRINT "B";
50 GOSUB 110
60 PRINT "C";
70 PRINT "D";
80 GOSUB 130
90 PRINT "E";
100 GOTO 10
110 PRINT "F";
120 RETURN
130 PRINT "G";
140 RETURN
150 PRINT "H";
160 END
```
- c) 

```
10 PRINT "A";
20 I = 3
30 ON I GOSUB 100,200,300,400
40 PRINT "B";
50 GOTO 30
100 PRINT "C";
110 I = 2
120 RETURN
200 PRINT "D";
210 I = 3
220 GOSUB 300
230 PRINT "E";
240 I = 4
250 RETURN
300 PRINT "F";
310 I = 1
320 RETURN
400 PRINT "G";
410 END
```
- d) 

```
10 PRINT "A";
20 GOSUB 100
30 PRINT "B";
40 GOSUB 200
50 PRINT "C";
100 PRINT "D";
110 RETURN
200 PRINT "E";
210 RETURN
```

Part II Determine what line 70 should be:

```
10 INPUT "ENTER YOUR NAME";N$
20 PRINT
30 PRINT "1. DELAY IT 10 - SECONDS"
40 PRINT "2. ENDLESS LOOP IT"
50 PRINT "3. FORGET IT"
60 INPUT "SELECTION";I
70
80 PRINT N$
90 GOTO 80
100 T = TIME
110 IF INT(TIME - T) <> 600 THEN 110
120 PRINT N$
130 END
```

# Worksheet 5.1 Solutions

## Part I

- a) ADBEC
- b) ABFCEDGEADGEAH
- c) AFBCBDFEBG
- d) ADBECD followed by ?RETURN WITHOUT GOSUB ERROR IN 110

## Part II

70 ON I GOTO 100,80,130

# Lesson 5.2

- OBJECTIVES: a) to develop strategies for designing successful, well-structured programs, including how to:
- i. outline programs using algorithms
  - ii. structure programs using modules and subroutines
  - iii. document programs using REM statements
- 

ASSIGNMENT: Read pages 5.5 through 5.11  
Text problems 7,9,11      Homework problems 8,12,20

---

Clear style and structure are every bit as important in a computer program as they are in an English essay. As programs become very long, they become considerably more complicated and therefore require careful planning. For this reason, it is imperative in teaching programming to insist on good style.

After completing the first five chapters, a student will have learned most of the BASIC commands necessary to write almost any program. At this point in your programming course, it is well worth spending a considerable amount of time discussing problem solving techniques and programming style. The major problem in holding such a discussion is that to this point students have not written programs long enough to see the value of developing such skills. You should convince your students that learning how to solve problems is a major objective of an introductory programming course and that its value will become obvious as problems of increasing difficulty are confronted.

Students tend to attack problems directly without first planning, an approach which should be strongly discouraged. It is a good practice to require students to submit to you a carefully outlined solution before beginning to program on the computer.

The text outlines the solution to a problem for writing a program to quiz students in mathematics. This lesson will demonstrate how to plan a solution to a problem of somewhat greater complexity.

The correct method for planning the solution to a lengthy problem is called "top-down" planning. This method involves working on the major portions of the solution first and leaving the specifics until later. The problem is first broken down into large blocks and the details of the blocks are filled in at a later time. We will use subroutines to code each of the large blocks.

The problem we will solve involves computing the payroll for a small company which employs five people. Each employee receives a weekly paycheck computed as follows:

1. The gross (total) pay is based on the employee's hourly wage times the hours worked; for every hour over 40 hours, the employee receives 1.5 times the hourly wage; over 50 hours, 2 times the hourly wage.
2. Taxes are deducted based on the following schedule:
  - \$200. - 10%
  - \$300. - 15%
  - \$400. - 25%
  - \$600. - 50%
3. Social security is deducted at 6.25% of the gross salary.
4. Participants in the company pension plan can have 10% of their gross wage subtracted and saved, but not all employees have elected to join the pension plan.
5. There is a vacation fund which allows employees to subtract from 1 to 5% of their gross wages.

First, we must determine what information the computer must have for each employee:

1. name
2. hourly wage
3. hours worked (which varies from week to week)
4. pension plan (yes or no, signified by 1 or 0)
5. vacation plan percentage.

This information would be best stored in a disk file, but since we have yet to cover disk files, we will store the information in data statements structured as follows:

```
900 DATA SMITH,4.25,1,5
```

The first string contains the employee's name, then the hourly wage, a number (0 or 1) for the pension plan, and a percentage for the vacation plan. Since the hours worked vary, they will be input from the keyboard when the program is run.

Next let's determine the large blocks by deciding what major tasks the program should perform.

1. Load information into arrays from the data statements and input the hourly wage from the keyboard.
2. Compute all deductions
  - (a) taxes and social security
  - (b) pension and vacation plans
3. Print out payroll.

We have three major tasks, with the second divided into two subdivisions. From this basic outline we can outline our program without filling in the specific details.

```
100 GOSUB 2000:REM LOAD ARRAY AND ACCEPT KEYBOARD INPUT
200 GOSUB 3000:REM DEDUCTIONS
300 GOSUB 4000:REM PRINT PAYROLL
```



```

10 REM
20 REM **PAY CHECK PROGRAM**
30 REM
40 REM -----
50 REM
60 REM *-VARIABLE DECLERATIONS
70 REM
80 REM N$( ) - EMPLOYEES NAME
90 REM S( ) - HOURLY WAGE
100 REM
110 REM D( ) - DEDUCTION CODE
120 REM --- 0 : JUST TAXES & SOC. SEC.
130 REM --- 1 : ALSO PENS. & VAC. FUND
140 REM
150 REM A( ) - % TO GO INTO VAC. FUND
160 REM
170 REM W( ) - HOURS WORKED
180 REM
190 REM G( ) - EMPLOYEE'S GROSS PAY
200 REM
210 REM T( ) - TAX DEDUCT
220 REM SS( ) - SS DEDUCT
230 REM F( ) - PENSION DEDUCT
240 REM V( ) - VAC. FUND
250 REM
260 REM Q - TARGET NUMBER THAT IS SENT
270 REM - THE ROUNDING SUBROUTINE
280 REM
290 REM ROUND - THE ROUNDED # RETURNED
300 REM
310 REM -----
320 REM
1000 REM *** MAIN BODY ***
1010 REM
1020 DIM N$(5),S(5),D(5),A(5)
1030 DIM T(5),SS(5),F(5),V(5),W(5)
1040 GOSUB 2000:REM LOAD ARRAY
1050 FOR I=1 TO 5
1060   GOSUB 3000:REM DEDUCTIONS
1070   PRINT
1080 NEXT I
1090 GOSUB 4000:REM PRINT OUT
1100 END
1110 REM *** END OFF MAIN BODY ***
1120 REM

```

```

2000 REM *** SUBROUTINE: LOAD ARRAY ***
2010 REM
2020 FOR I=1 TO 5
2030 PRINT
2040 READ N$(I),S(I),D(I),A(I)
2050 DATA BIDWELL,7.65,1,4
2060 DATA WANG,3.35,0,0
2070 DATA ZAHARCHUK,5,0,0
2080 DATA DINGLE,5,1,2
2090 DATA NILSON,6.45,1,3
2100 GOSUB 5000:REM HOURS WORKED
2110 GOSUB 6000:REM GROSS PAY
2120 NEXT I
2130 REM
2140 REM LOAD TAX PERCENT ARRAY
2150 REM
2160 FOR I=2 TO 6
2170 READ TR(I)
2180 NEXT I
2190 DATA .1,.15,.25,.25,.3
2200 RETURN
2210 REM *** END OF LOAD ARRAY ***
2220 REM
3000 REM *** SUBROUTINE: DEDUCTIONS ***
3010 REM
3020 REM TAXES & SOC. SECURITY
3030 REM
3040 M=0
3050 M=M+1
3060 IF INT(.01*G(I))=M THEN 3090
3070 IF M<6 THEN 3050
3080 IF M<2 THEN 3130
3090 P=TR(M)
3100 T(I)=P*G(I)
3110 Q=T(I):GOSUB 7000:REM ROUNDING
3120 T(I)=ROUND
3130 SS(I)=.0625*G(I)
3140 Q=SS(I):GOSUB 7000:REM ROUNDING
3150 SS(I)=ROUND
3160 IF NOT D(I) THEN RETURN
3170 REM
3180 REM PENSION PLAN AND VACATION FUND
3190 REM
3200 P(I)=.1*G(I)
3210 Q=P(I):GOSUB 7000:REM ROUNDING
3220 P(I)=ROUND
3230 V(I)=(A(I)/100)*G(I)
3240 Q=V(I):GOSUB 7000:REM ROUNDING
3250 V(I)=ROUND
3260 RETURN
3270 REM *** END OF DEDUCTIONS ***
3280 REM

```

```

4000 REM *** SUBROUTINE: PRINT OUT ***
4010 REM
4020 FOR I=1 TO 5
4030 PRINT
4040 PRINT "EMPLOYEE NO. ";I
4050 PRINT
4060 PRINT "NAME ==>> ";N$(I)
4070 PRINT "PAY RATE ==>> $";S(I);"/HR"
4080 PRINT "HOURS WORKED ==>> ";W(I)
4090 PRINT "GROSS PAY ==>> $";G(I)
4100 PRINT "DEDUCTIONS"
4110 PRINT "    TAXES          $";T(I)
4120 PRINT "    SS DEDUCT      $";SS(I)
4130 PRINT "    PENSION        $";P(I)
4140 PRINT "    VAC. FUND     $";V(I)
4150 AMOUNT = G(I)-(T(I)+SS(I)+P(I)+V(I))
4160 PRINT
4170 PRINT "PAYCHECK ==>> ";AMOUNT
4180 PRINT
4190 NEXT I
4200 RETURN
4210 REM
4220 REM *** END OF PRINT OUT ***
4230 REM
4240 REM
4250 REM
4260 REM
5000 REM ***SUBROUTINE: HOURS WORKED ***
5010 REM
5020 PRINT "EMPLOYEE #";I;" ";N$(I)
5030 INPUT "HOW MANY HOURS DID HE WORK ==>> ";W(I)
5040 PRINT
5050 IF W(I)<0 OR W(I)>1000 THEN PRINT "REENTER...":GOTO 5020
5060 RETURN
5070 REM *** END OF HOURS WORKED ***
5080 REM
6000 REM ***SUBROUTINE: CALCULATE GROSS PAY ***
6010 REM
6020 IF W(I)<40 THEN G(I)=W(I)*S(I):GOTO 6050
6040 G(I)=40*S(I) + (1.5*S(I)*(W(I)-40))
6050 Q=G(I):GOSUB 7000:REM ROUNDING
6060 G(I)=ROUND
6070 PRINT "THIS EMPLOYEE'S GROSS PAY IS $";G(I)
6080 PRINT "-----"
6090 RETURN
6100 REM *** END OF CALCULATE GROSS PAY ***
6110 REM
7000 REM *** SUBROUTINE: ROUNDING ***
7010 REM
7020 ROUND=INT(100*Q+.5)/100
7030 RETURN
7040 REM *** END OF ROUNDING ***

```

A sample run of the above program follows, as well as a schematic diagram of the program's structure and hierarchy:

RUN

EMPLOYEE #1 BIDWELL  
 HOW MANY HOURS DID HE WORK ==>> 43  
  
 THIS EMPLOYEE'S GROSS PAY IS \$340.43

•  
•  
•

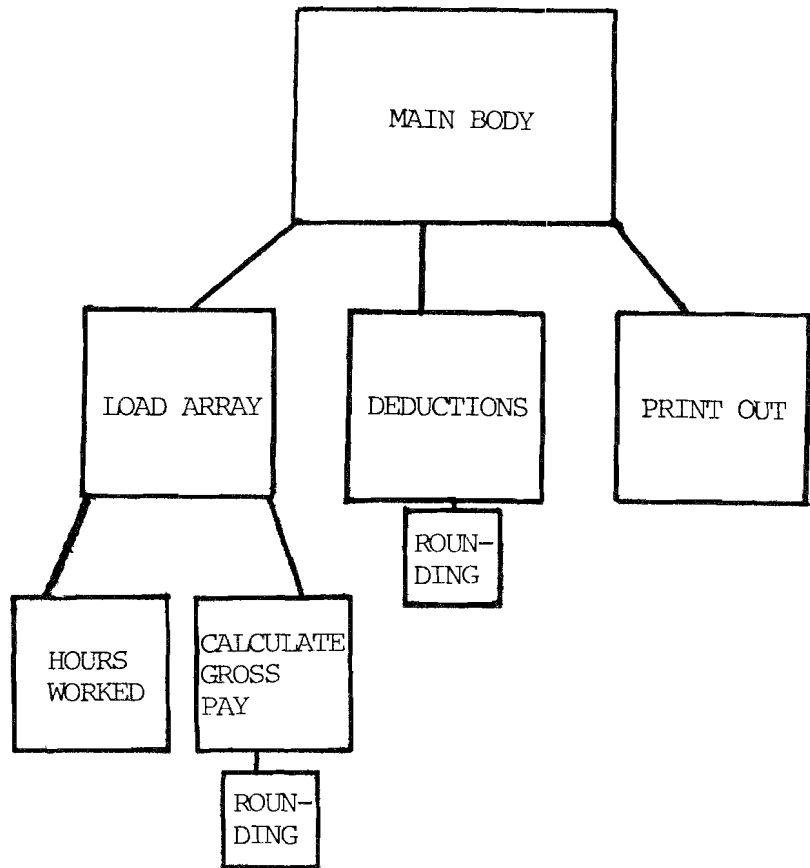
EMPLOYEE #5 NILSON  
 HOW MANY HOURS DID HE WORK ==>> 56  
  
 THIS EMPLOYEE'S GROSS PAY IS \$432.15

•  
•  
•

EMPLOYEE NO. 1  
  
 NAME ==>> BIDWELL  
 PAY RATE ==>> \$7.65/HR  
 HOURS WORKED ==>> 43  
 GROSS PAY ==>> \$340.43  
 DEDUCTIONS  
   TAXES           \$51.06  
   SS DEDUCT       \$21.28  
   PENSION         \$34.04  
   VAC. FUND       \$13.62  
  
 PAY CHECK ==>> \$220.43

•  
•  
•

EMPLOYEE NO. 5  
  
 NAME ==>> NILSON  
 PAY RATE ==>> \$6.45/HR  
 HOURS WORKED ==>> 56  
 GROSS PAY ==>> \$432.15  
 DEDUCTIONS  
   TAXES           \$108.04  
   SS DEDUCT       \$27.01  
   PENSION         \$43.22  
   VAC. FUND       \$12.96  
  
 PAY CHECK ==>> \$240.92



In any long, complex program that uses many variables, a variable declaration section, which declares all of the variables that are to be used and what they represent, can help explain the purpose and the method of the program. Lines 10 through 320 comprise the variable declaration section for the above program.

The main body of this program consists only of the three calls to the subroutines that were discussed earlier. The REM statements at lines 1000 and 1110 signal the beginning and the end of the main body, respectively.

The subroutine called LOAD ARRAY, called from line 1040, starts at line 2000 and ends at line 2210. It performs several functions. First, it fills in the employees' names, wages and deduction packages, using the READ..DATA statements that occur in lines 2040 to 2090. After this data is loaded, two additional functions are used. Line 2100 calls a subroutine HOURS WORKED, which starts at line 5000. This subroutine allows the user to input the hours worked by each employee and tests the input for undesirable inputs. One subroutine calling another subroutine is legal and can lend clarity to a program's statement. If at a later time it becomes necessary to change the program, locating the specific lines to be edited is made easier by this structure.

Line 2110 also calls a subroutine, called CALCULATE GROSS PAY, which, as the name suggests, calculates each employee's gross (i.e. before deductions) pay for the week, using each employee's hourly salary and the number of hours worked. These two variables are used to calculate the gross pay using the formula outlined in the problem description.

Lines 2130 to 2190 fill an array which will be used later to calculate the amount that must be deducted from each pay check because of tax. Each subscript represents \$100 of gross pay (i.e. TR(4) is the percentage tax you must pay if your gross pay is between 400 and 499 dollars). After this brief function the program RETURNS (line 2200) to the main body.

Next, the subroutine DEDUCTIONS is called from line 1060. This subroutine calculates the various deductions that must be subtracted from the employee's gross pay. The deduction due to taxes is calculated between lines 3040 and 3110, and the amount of each is stored in the variable T(I). After figuring the social security deduction, the program uses a conditional statement to determine whether to deduct for pension and vacation plans. The subroutine ends on line 3260 and RETURNS to the main body. (Note: This procedure is repeated for each employee separately.)

Finally, the program calls the last of the three subroutines, PRINT OUT, from line 1090, which prints out a detailed report on each employee, using the information that was obtained in the previous two subroutines. After printing out the pay reports, the program RETURNS to line 1100, where it ends.

An excellent example of the usefulness of a subroutine is the subroutine ROUNDING which, although only one line long, is called from throughout the program to round all calculations to the nearest cent.

We believe the best way to plan a computer program is by writing out a plan (algorithm) as we have done in this lesson. Some teachers, however, also believe in flowcharting, so we have included a section from an earlier publication as an introduction to that technique of planning a computer program.

## FLOWCHARTING

As programs become more complex, it becomes especially easy to forget important details or to lose track of the direction that the program must take. Such problems can be avoided by preparing an outline in the form of a flow chart before the program is actually written.

### Symbols for Flowcharting

A large program often consists of two parts; the main body of the program in one, and its functions and subroutines in the other. Usually the main body comes first and the functions and subroutines second. Execution of the main body should flow from top to bottom. Any GOTO statements should send execution to a point before the GOTO, and not after it. This format makes a program easier to be read and, in many cases, more efficient. To write such a program the programmer must first have a comprehensible outline. One way to construct an outline is to draw what is called a flow chart. A flow chart consists of various symbols and words which graphically show what the program is to do. Because of their pictorial form, flow charts make potential deficiencies in a program easy to spot.

There are five different symbols in flowcharting. The first symbol is



This is a terminator which indicates the beginning or end of a program or subroutine. The

second symbol is



and is called the Input/Output or I/O symbol. This

indicates that data is either entering or leaving the program. Generally, it designates an operation with the disk or tape, or an INPUT, READ, or PRINT statement. The following is a flow chart to input a number (X) and print the number multiplied by three.



One possible program for this flow chart would be the following:

```
10 INPUT X
20 PRINT 3*X
```

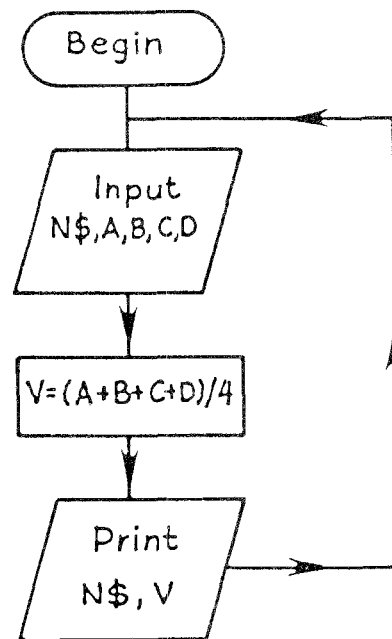
Note the arrows in the flow chart which indicate the direction of flow in the program. Also note that the flow chart proceeds from left to right. In general, flow charts proceed from left to right or from top to bottom. In more complex examples, they may move both from left to right and from top to bottom.

The next symbol is the process symbol which indicates the processing of some data



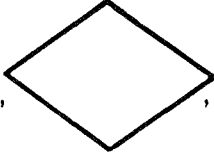
. It is the most general of the five symbols and is used whenever one of

the other four more specific symbols does not apply. The following is a flow chart for a program which allows the user to input a student's name and four course grades. It computes the average of the grades (process symbol), prints the student's name and average, and then returns to begin the cycle again. The accompanying program was written based upon the flow chart.

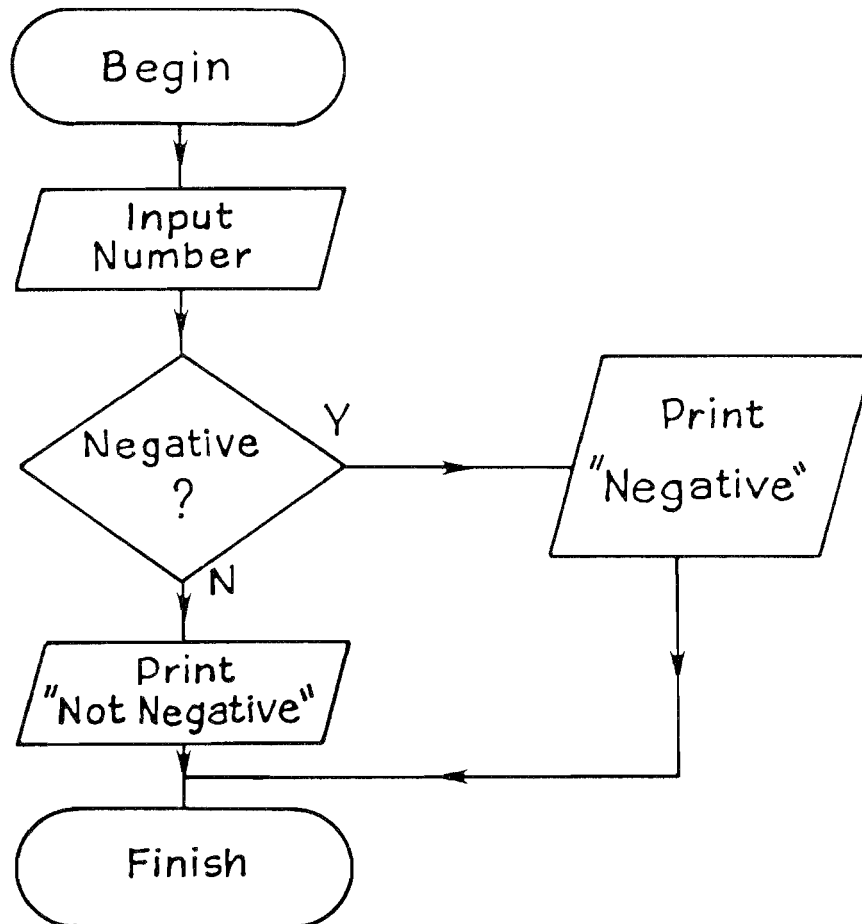


```
10 INPUT "NAME AND 4 GRADES: ";N$,A,B,C,D
20 V = (A + B + C + D) / 4
30 PRINT N$,V
40 GOTO 10
```



The fourth symbol, , designates a conditional, indicating that a decision

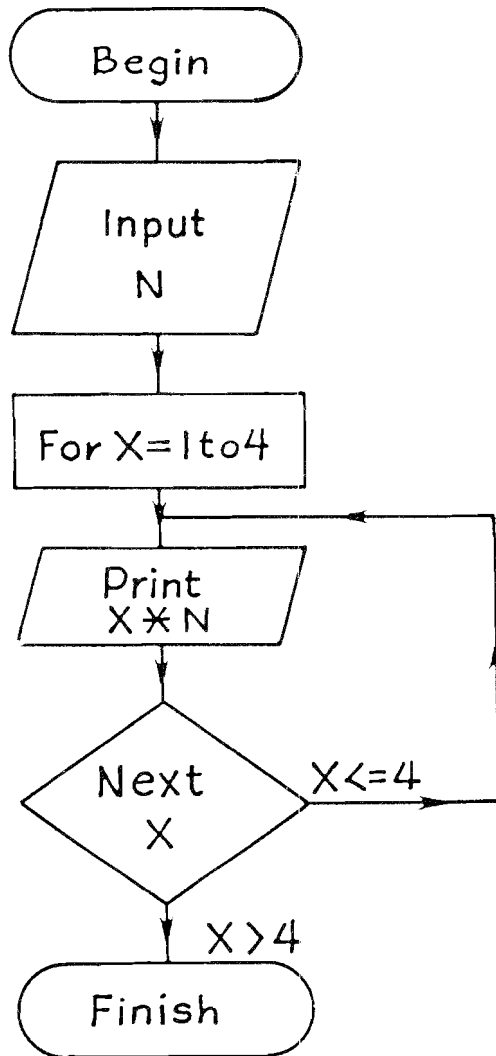
is to be made. The IF . . . THEN statement is one type of conditional. The following flow chart specifies a number that is input and then goes on to determine if it be negative or not.



```
10 INPUT X
20 IF X < 0 THEN PRINT "NEGATIVE": END
30 PRINT "NOT NEGATIVE"
```

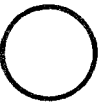
The conditional symbol always has two or more arrows coming from it. In this case there are two such arrows, labeled Y for YES and N for NO.

The NEXT X portion of a FOR . . . TO, NEXT loop is also a conditional. The flow chart and accompanying program ask the user to Input a number and then print four multiples of the number.

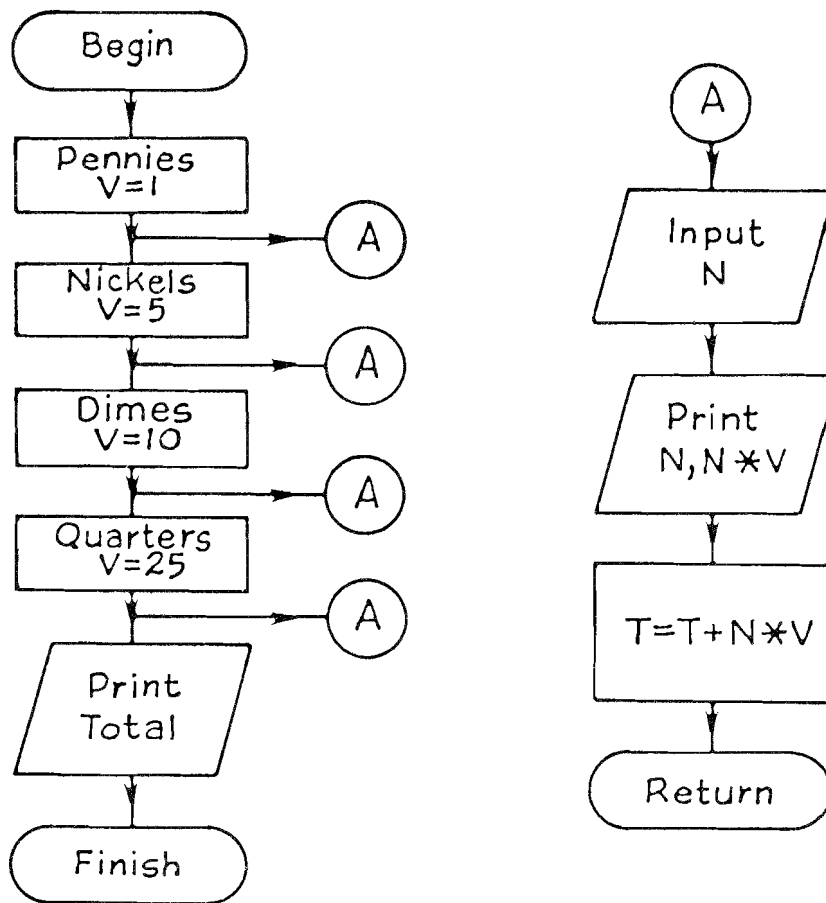


```
10 INPUT N
20 FOR X = 1 TO 4
30 PRINT X * N
40 NEXT X
```

Each time the conditional symbol is reached, X is incremented and a decision is made based on its new value. If X is 4 or less, the loop continues and another result is printed; if X exceeds 4, the run ends. The two exit symbols from the conditional allow these two possibilities.

The last symbol is the connector symbol, which is a circle: . It is used to connect separate parts of a flow chart. One important way in which this symbol is used is to indicate the connections between the main body of a program and its subroutines. Letters are placed in the connector symbols to indicate which connect with which.

The following is a flow chart for Program 7.1.



When a connector symbol is encountered in the main program, execution proceeds to the connector bearing the same letter in the subroutine. You should understand that after the RETURN box is reached, execution goes back to the box in the main program which follows the last connector used.



# Worksheet 5.2

Part I Write a program incorporating subroutines to perform major tasks. The program will play the game 'TRIVIA' and will ask the user to answer a series of questions. The program may not repeat a question during the game. The program should print the score at the end of the game. Use the following READ...DATA statements in your program:

```
10000 DATA 10: REM NUMBER OF Q
      QUESTIONS
10010 DATA "HOW MANY HOMERS DID
      BABE RUTH HIT?","714"
10020 DATA "WHERE IS THE GRAND C
      ANYON?","ARIZONA"
10030 DATA "WHAT IS PI TO TWO DE
      CIMAL DIGITS?","3.14"
10040 DATA "WHO DISCOVERED THE T
      HEORY OF RELATIVITY?","EINST
      EIN" .
10050 DATA "WHERE IS THE SPHINX
      LOCATED?","GAZA"
10060 DATA "WHO WON THE 1974 STA
      NLEY CUP?","PHILADELPIA FLYE
      RS"
10070 DATA "WHAT'S THE CAPITAL
      OF NEW JERSEY?","TRENTON"
10080 DATA "WHAT'S THE CAPITAL O
      F SPAIN?","MADRID"
10090 DATA "WHO WON THE DECATHA
      LON IN 1976?" , "BR
      UCE JENNER"
10100 DATA "WHAT IS THE SPEED OF
      LIGHT?","3E8"
```

The RUN of the program should appear as follows:

```
      RUN
1> WHAT IS THE SPEED OF LIGHT?
===>>3E8
CORRECT

2> WHO WON THE DECATHALON IN 1976?
===>>BRUCE JENNER
CORRECT

3> WHAT'S THE CAPITAL OF NEW JERSEY?
===>>PRINCETON
NO, THE CORRECT ANSWER WAS: TRENTON
```

# Worksheet 5.2 Solutions

```
Part I 10 REM *** MAIN BODY ***
20 REM
30 GOSUB 1000:REM CREATE QUESTION & ANSWER LISTS
40 FOR I=1 TO MX
50   GOSUB 3000:REM QUESTION AND REPLY
60   GOSUB 4000:REM KEEP SCORE
70 NEXT I
99 END
100 REM
110 REM
1000 REM CREATE RANDOM QUESTION & ANSWER LISTS
1010 REM
1020 DIM Q$(10):DIM A$(10):REM ARRAY SIZE SHOULD BE THE SAME AS MX
1030 READ MX
1040 FOR I=1 TO MX
1050   READ TQ$,TA$
1060   GOSUB 2000:REM LOCATE RANDOM EMPTY POSITION
1070   Q$(PL)=TQ$:A$(PL)=TA$
1080 NEXT I
1090 RETURN
1100 REM
1110 REM
2000 REM RANDOM POSITION GENERATOR
2010 REM
2020 PL=INT(RND(0)*MX+1)
2030 IF Q$(PL)<>" " THEN 2020
2040 RETURN
2050 REM
2060 REM
3000 REM QUESTION AND REPLY
3010 REM
3020 PRINT I;"> ";Q$(I)
3030 INPUT "===>";T$
3040 RL$="CORRECT"
3050 IF T$<>A$(I) THEN RL$="NO, THE CORRECT ANSWER WAS:"+A$(I)
3060 PRINT RL$:PRINT
3070 RETURN
3080 REM
3090 REM
4000 REM SCORE ROUTINE
4010 REM
4020 IF RL$="CORRECT" THEN SC=SC+1
4030 IF I=MX THEN PRINT "SCORE:";SC
4040 RETURN
4050 REM
4060 REM
10000 DATA 10:REM NUMBER OF QUESTIONS
10010 DATA "HOW MANY HOMERS DID BABE RUTH HIT?","714"
10020 DATA "WHERE US THE GRAND CANYON?","ARIZONA"
10030 DATA "WHAT IS PI TO TWO DECIMAL DIGITS?","3.14"
10040 DATA "WHO DISCOVERED THE THEORY OF RELATIVITY?","EINSTEIN"
10050 DATA "WHERE IS THE SPHINX LOCATED?","GAZA"
10060 DATA "WHO WON THE 1974 STANLEY CUP?","PHILADELPHIA FLYERS"
10070 DATA "WHAT'S THE CAPITOL OF NEW JERSEY?","TRENTON"
10080 DATA "WHAT'S THE CAPITOL OF SPAIN?","MADRID"
10090 DATA "WHO WON THE DECATHALON IN 1976?","BRUCE JENNER"
10100 DATA "WHAT IS THE SPEED OF LIGHT?","3E8"
```

# Lesson 5.3

- OBJECTIVES: a) to discriminate between runtime (syntactical and semantic) errors and logic errors
- b) to demonstrate the importance of hand-tracing for program development and debugging
- 

ASSIGNMENT: Read pages 5.12 through 5.20 Do review problems 4a,4b  
Text problems 5,13,15,17,19 Homework problems 6,14,16,18

---

The first debugging tool which a student should master is the technique called 'hand tracing'. The worksheets in this manual have emphasized hand tracing (e.g., exact output problems).

See if students can agree on the output of the following program:

```
10 A = 2
20 B = -2
30 FOR C = 1 TO 4
40   A = A + B + C
50   B = B + A - C
60   PRINT A,B
70 NEXT C
80 END
```

A common error students make is to take A as 2 the first time line 50 is executed. You must remind students that a variable can only have one value at a time: the one most recently assigned. This can be emphasized by using the familiar variable boxes. Each time a variable value is altered, erase the old value and write in a new. For the sample program above, the boxes would appear as follows just prior to execution of line 50.

A	B	C
1	-2	1

The result of line 50 is now clearly -2.

Your students will find it most instructive if you use the boxes and the following chart simultaneously. The final result for the program above will be as follows:

A	B	C
Ø	-9	5

LINE	A	B	C	OUTPUT
	0	0	0	
10	2			
20		-2		
30			1	
40	1			
50		-2		
60				1,2
70 → 30			2	
40	1			
50		-3		
60				1,-3
70 → 30			3	
40	1			
50		-5		
60				1,-5
70 → 30			4	
40	0			
50		-9		
60				0,-9
70 → 30			5	

The table will always begin with zeroes. Also, in this program, the final value of C will be 5. (Do your students remember why?)

Students may desire to cross off the 'old' values so that the 'most recent' value is always clearly evident. If your students find this sample easy, challenge them with the worksheet.

A debugging technique not specifically mentioned in the text is the simple immediate mode PRINT. Variable values are retained in their respective boxes following program execution and are therefore available for inspection. As an example of this technique's usefulness, consider the following program which (among other things) should count slowly from 1 to 10:

```

10  FOR I = 1 TO 10
20  PRINT I
30  REM
40  REM
50  REM
60  FOR I = 1 TO 2500
70  NEXT I
80  REM
90  REM
100 REM
110 NEXT I

RUN
1

?NEXT WITHOUT FOR ERROR ON 110
PRINT I
2501

```

This information revealed by the immediate mode PRINT should help the programmer notice that the index I was mistakingly used in the delay loop at line 60.



# Worksheet 5.3

Part I Use the table to hand trace the output of the following program:

```
10 A = 2
20 X = X + Y + A
30 FOR J = 1 TO X
40   Y = Y * X + 1
50 NEXT J
60 IF A < 0 THEN 80
70 A = -2: GOTO 20
80 PRINT X,Y
90 END
```

A	X	Y

Part II Locate and correct the logic error in the following program:

```
10 REM AVERAGE 5 NUMBERS
20 FOR I = 1 TO 5
30   INPUT X
40   S = S + X
50 NEXT I
60 PRINT "THE AVERAGE IS";S / 5
70 INPUT "TRY AGAIN (Y/N)?";A$
80 IF A$="Y" THEN 20
90 END
```

Part III Determine the immediate mode output missing from the following sequence (i.e. fill in the boxes):

```
10 FOR I = 1 TO 5
20   READ X$(I),X$(I - 1)
30   IF I = 3 THEN STOP
40 NEXT I
50 DATA APE,BAT,CAT,DUCK,EEL,FISH,GNAT,HARE,LLAMA,MOOSE
```

RUN

```
BREAK IN 30
PRINT X$(I)
```

CONT

```
PRINT X$(I)
```

# Worksheet 5.3 Solutions

Part I

A	X	Y
$\emptyset$	$\emptyset$	$\emptyset$
<del>2</del>	<del>2</del>	<del>1</del>
<del>-2</del>	<del>3</del>	<del>3</del>
		10
		31

RUN  
3            31

Part II The program executes perfectly the first time through. However, if the user selects 'Y' at line 70, a logic error ensues. The sum S in line 40 continues to increase beyond the total of the first five numbers. The variable must be 'reset' by changing line 80 to

```
80 IF A$ = "Y" THEN 15
```

and adding

```
15 S = 0
```

Part III EEL

<BLANK>

# Chapter 5 Supplementary Problems

- 1) Design and debug a program which simulates the movement of a prince lost in the darkness of Castle Dracula. Use the following specifications:

## The Board

- a) The castle floor is made of a 10 \* 10 matrix.
- b) There is an invisible wall around the castle which can not be passed through (check for subscript out of range).
- c) Randomly place 2 piles of treasure on the castle floor.
- d) Randomly place 3 trap doors on the castle floor.
- e) Randomly place 1 escape door on the castle floor.
- f) Assign the prince the starting position of (2,3).
- g) Assign any empty elements to the value "\*". (see figure 1)

## The Game

The object of the game is to collect both treasure piles and leave via the escape door. After a move replace the value of the old position with a "\*" and the new position with a "P". Remember to check the value of the new square before replacing it.

A player may move north, south, east or west and has the option of lighting a match to see the adjacent positions (see figure 2). He is limited to 5 matches. If a player moves onto a trap door, he perishes and the program is over.

### A List of Board Values

\* = empty  
T = treasure  
D = trap door  
E = escape door  
P = prince

### A List of Variables

B\$(x,y) = Board positions  
M = matches  
T\$ = temporary (hold values)  
X = row  
Y = column  
T = treasure

When the prince finally exits the castle, print out the appropriate message.

TRAP DOOR! YOU ARE EATEN BY CROCODILES.  
PASSAGE TO OUTSIDE! YOU FOUND NO TREASURES, SORRY.  
PASSAGE TO OUTSIDE! AMATEUR - YOU FOUND 1 PILE OF TREASURE  
PASSAGE TO OUTSIDE! PRO - YOU FOUND BOTH PILES OF TREASURE

Above all, be sure to use the programming techniques set forth in Chapter 5 when attempting this program. Your program should be neat and the logic should be easy to follow. If your program contains many errors and is not well-structured, you will find that debugging will take a long time.

*	*	*	*	*	*	*	*	*
*	*	P	*	*	*	*	*	*
*	D	*	*	*	*	*	T	*
*	*	*	*	*	*	*	*	*
*	*	*	*	*	*	*	*	*
*	*	*	*	*	E	*	*	*
*	*	*	D	*	*	*	*	*
*	*	*	*	*	*	*	*	D
*	*	T	*	*	*	*	*	*
*	*	*	*	*	*	*	*	*

fig 1

*	*	*
*	P	*
D	*	*

fig 2

2) The following programs contain errors, including at least one logic error each. Find the errors and correct them:

```
a)  10  REM CREATE A LIST OF 20 RANDOM
     20  REM NUMBERS BETWEEN 1 AND 100
     30  REM WITH NO DUPLICATES
     40  FOR I = 1 TO 50
     50  T = (INT ( RND (1) * 100) + 1
     60  FOR J = 1 TO I
     70  IF T = X(J) THEN 20
     80  X(I) = T
     90  REM PRINT LIST
    100  FOR J = 1 TO T
    110  PRINT X(J)
    120  NEXT J
```

```
b)  10  REM ALLOW A CLUB 'BOUNCER' TO
     20  REM VERIFY THAT A PERSON REQUESTING
     30  REM ADMISSION ACTUALLY BELONGS TO
     40  REM A VERY EXCLUSIVE CLUB.
     50  INPUT "INITIALS OF PERSON REQUESTING ADMISSION"?;ADMS$
     60  FOR I = 1 TO 20
     70    READ VIP$
     80    IF ADMS$ = VIP$ THEN PRINT "O.K. LET;ADMS$;"ENTER"
     90  NEXT I
    100  PRINT "NOT ALLOWED IN"; GOTO 50
    110  DATA JCE,JML,MAL,EJE,YDE,MFE,AFC,RGE,ELW,JRD,LPE,BEM
         PPP,SKB,KFW,SDB,PDQ,APP,JDF
```

3) Write a program to move a knight randomly on a chess board (8x8). The knight cannot go off the board. Use a RND statement to get a random number between 1 and 8. Use ON...GOSUB to check and plot the knight's new position. Count the number of moves the knight takes and print out the board with the number of each landing point in its appropriate square. Start the knight at the point (4,4) with starting value of 1. If the computer picks 10 invalid moves in a row, consider the simulation over.

# Chapter 5 Supplementary Problem Solutions

```

1) 10 DIM B$(11,11)
20 REM FILL THE GRID WITH ASTERISKS
30 REM
40 FOR X=1 TO 10
50   FOR Y=1 TO 10
60     B$(X,Y)="*"
70   NEXT Y
80 NEXT X
90 REM
100 REM POSITION TRAP DOORS
110 FOR X=1 TO 6
120   R(X) = INT(10*RND(0))+1
130   C(X) = INT(10*RND(0))+1
140   IF X=1 THEN 190
150   FOR I=1 TO X-1
160     IF R(X)=R(I) AND C(X)=C(I) THEN 120
170   NEXT I
180   READ F$
190   B$(R(X),C(X))=F$
200 NEXT X
210 DATA "T","T","D","D","D","E"
220 R1=2:C1=3
230 SF=1:GOTO 4000
1000 PRINT "YOU ARE ";S$;" AT ROW";R1;" COL";C1
1010 PRINT "YOU HAVE COLLECTED ";T;" TREASURE(S)"
1020 PRINT "WHERE WOULD YOU LIKE TO MOVE"
1030 PRINT
1040 PRINT "(1) NORTH"
1050 PRINT "(2) SOUTH"
1060 PRINT "(3) EAST"
1070 PRINT "(4) WEST"
1080 PRINT "(5) LIGHT A MATCH"
1090 PRINT
1100 PRINT "WHERE TO, BOSS ==>";
1110 INPUT D
1120 PRINT "CHOICE=";D
1130 IF D.1 OR D>5 THEN 1030
1140 ON D GOSUB 2000,2010,2020,2030,3000
1150 GOTO 4000
2000 RI=-1:CI=0:RETURN
2010 RI=1:CI=0:RETURN
2020 RI=0:CI=1:RETURN
2030 RI=0:CI=-1:RETURN
3000 PRINT:IF M>4 THEN PRINT "OUT OF MATCHES...":PRINT:S$="STILL":GOTO 1000
3010 FOR R=-1 TO 1
3020   FOR C=-1 TO 1
3030     IF R=0 AND C=0 THEN PRINT "P ";GOTO 3060
3050     PRINT B$(R1+R,C1+C);" ";
3060   NEXT C
3070 PRINT
3080 NEXT R
3090 PRINT
3100 M=M+1
3110 PRINT "YOU HAVE ";5-M;" MATCH(ES) LEFT"
3120 PRINT:GOTO 1000
4000 IF C1+C1>10 OR C1+C1<1 THEN 5130
4010 IF R1+R1>10 OR R1+R1<1 THEN 5130
4020 S$="NOW"
4030 REM
4040 REM LEGAL MOVE
4050 REM
4060 R1=R1+R1:C1=C1+C1
5000 REM
5010 REM CHECK SPOT
5020 REM
5030 REM
5040 IF B$(R1,C1)="T" THEN T=T+1:B$(R1,C1)="*":PRINT"*TREASURE*":PRINT:GOTO 1000
5050 IF B$(R1,C1)="D" THEN PRINT "TRAP DOOR! YOU ARE EATEN BY CROCODILES.":END
5060 IF B$(R1,C1)="E" THEN PRINT "PASSAGE TO OUTSIDE! ":GOTO 5030
5070 PRINT "NOTHING SPECTACULAR HERE, BOSS":PRINT:GOTO 1000
5080 ON T+1 GOSUB 5100,5110,5120
5090 PRINT:END
5100 PRINT "NO TREASURE FOUND. SORRY.":RETURN
5110 PRINT "YOU FOUND A SINGLE PIECE OF TREASURE":RETURN
5120 PRINT "YOU FOUND FOUR PIECES OF TREASURE":RETURN
5130 IF T=1 THEN T="**" ELSE T="***":PRINT:GOTO 1000

```

2) The corrected lines should look like this:

```
a) 40 FOR I = 1 TO 20
    50 T = INT (RND(0) * 100) + 1
    60 J = 1
    65 IF J <> I - 1 THEN J = J + 1: GOTO 80
    65 IF I = 1 THEN 80
    70 IF T = X(J) THEN 85
    85 NEXT I

b) 50 INPUT "INITIALS OF PERSON REQUESTING ADMISSION ";ADM$
    55 RESTORE
    60 FOR I = 1 TO 19
    80 IF ADM$ = VIP$ THEN PRINT "O.K. LET ";ADM$;" ENTER"
110 DATA JCE,JML,MAL,EJE,YDE,MFE,AFC,RGE,ELW,JRD,LPE,BBM,
    PPP,SKB,KFW,SDB,PDQ,APP,JDF
```

```

3) 10 REM SUPPLEMENTARY PROBLEM #3
20 REM      A KNIGHT'S TOUR
30 REM
40 DIM B(9,9)
50 M=1
60 GOSUB 580
70 IF W=25 THEN GOTO 130
80 REM GET A RANDOM NUMBER TO SELECT
90 REM      A MOVE
100 R=INT(RND(0)*9)+1
110 ON R GOSUB 160,210,260,310,350,410,460,510
120 GOTO 70
130 GOSUB 710
140 END
150 REM MOVEMENT TYPE - 1
160 X1=X+1:Y1=Y+2
170 IF X1=9 OR Y1=9 THEN W=W+1:GOTO 70
180 GOSUB 840
190 RETURN
200 REM MOVEMENT TYPE - 2
210 X1=X+1:Y1=Y-2
220 IF X1=9 OR Y1=1 THEN W=W+1:GOTO 70
230 GOSUB 840
240 RETURN
250 REM MOVEMENT TYPE - 3
260 X1=X-1:Y1=Y+2
270 IF X1=1 OR Y1=9 THEN W=W+1:GOTO 70
280 GOSUB 840
290 RETURN
300 REM MOVEMENT TYPE - 4
310 X1=X-1:Y1=Y-2
320 IF X1=1 OR Y1=1 THEN W=W+1:GOTO 70
330 GOSUB 840
340 RETURN
350 REM MOVEMENT TYPE - 5
360 X1=X-2:Y1=Y-1
370 IF X1=1 OR Y1=1 THEN W=W+1:GOTO 70
380 GOSUB 840
390 RETURN
400 REM MOVEMENT TYPE - 6
410 X1=X+2:Y1=Y-1
420 IF X1=9 OR Y1=1 THEN W=W+1:GOTO 70
430 GOSUB 840
440 RETURN
450 REM MOVEMENT TYPE - 7
460 X1=X+1:Y1=Y+1
470 IF X1=9 OR Y1=9 THEN W=W+1:GOTO 70
480 GOSUB 840
490 RETURN
500 REM MOVEMENT TYPE - 8
510 X1=X+1:Y1=Y+1
520 IF X1=9 OR Y1=9 THEN W=W+1:GOTO 70
530 GOSUB 840
540 RETURN
550 REM
560 REM BOARD SET-UP
570 REM
580 FOR X=1 TO 9
590   FOR Y=1 TO 9
600     B(X,Y)=0
610   NEXT Y
620 NEXT X
630 B(4,4)=M
640 M=M+1
650 W=0
660 X=4:Y=4
670 RETURN
680 REM
690 REM BOARD PRINT-OUT
700 REM
710 FOR I=1 TO 25:PRINT:NEXT I
720 FOR X=1 TO 9
730   Z=6
740   FOR Y=1 TO 9
750     PRINT TAB(Z);B(X,Y);
760     Z=Z+4
770   NEXT Y
780 PRINT:PRINT
790 NEXT X
800 RETURN
810 REM
820 REM MOVE/CHLDR ROUTINE
830 REM
840 IF B(X1,Y1)=10 THEN W=W+1:RETURN
850 X=X1:Y=Y1
860 B(X,Y)=M
870 M=M+1
880 RETURN

```



# Chapter 5 Test

Part I Determine the exact output of the following programs:

- a) 

```
10 X = 1
20 ON X GOTO 100,200,300,500
100 PRINT "A";
200 PRINT "B";
300 PRINT "C";
400 X = X + 1: GOTO 20
500 END
```
- b) 

```
10 I = 4
20 ON I GOSUB 50,60,40,70
30 PRINT "END": END
40 PRINT "A": RETURN
50 PRINT "B": RETURN
60 PRINT "C": RETURN
70 PRINT "D": RETURN
```
- c) 

```
10 X = 6
20 ON (X-3) GOSUB 50,60,70
30 IF X <> 0 THEN 20
40 PRINT "MY":END
50 PRINT "M";:X = 5:RETURN
60 PRINT "O";:X = 0:RETURN
70 PRINT "M";:X = 4:RETURN
```
- d) 

```
10 FOR V = 1 TO 4
20 READ A,B
30 IF B = 0 THEN STOP
40 S = (A + B) / B
50 PRINT A,B,S
60 NEXT V
70 DATA 2,1,4,2,6,3,8,0
```

Part II Circle the programming error(s) within each program. Explain what must be done to correct the error. If no error exists, state so.

- a) 

```
10 FOR J = 1 TO 4
20 READ I,J,K
30 PRINT I+J+K
40 NEXT J
50 DATA 7,3,19,201,6,28
60 DATA 0,117,42,37,201,5
```
- b) 

```
10 FOR J = 1 TO 4
20 READ I,J$,K
30 PRINT J$
40 NEXT J
50 DATA 2,SUGAR,9,8,SWEET,14,3
60 DATA 17,SOUR,5
```
- c) 

```
10 GOSUB 50
20 PRINT A$
30 GOSUB 100
40 PRINT B$
50 READ A$,B$
60 RETURN
70 DATA JOHN,BARB,BENNY,ALICE
```
- d) 

```
10 REM PRINT OUT A TABLE OF NAMES
20 REM AND PHONE NUMBERS
30 PRINT "NAME", "NUMBER"
40 PRINT "----", "-----"
50 FOR I = 1 TO 4
60 READ N$,T$
70 PRINT N$,T$
80 NEXT I
90 DATA "MARCI ETHER",777-6666,"FANNY FUTZPATRICK",666-5555
, "MARY JONES",555-4444,"BIRDIE SONG",444-6666
```

Part III Write each of the following programs as concisely as possible:

- a) Write a program which will attempt to find integer values of X and Y between -10 and 10 which satisfy the following equation:

$$\frac{X}{Y} + \frac{Y}{X} + \frac{1}{X+Y} = (X * Y) ^ 2$$

using STOP...CONT to prevent the program from ending in the event that a divide by zero error occurs. Make sure you are able to change the value of X or Y and continue with processing.

- b) Write a program which will have the user input a number that will represent a yearly income for an employee of XYZ Manufacturing. Using subroutines rather than IF...THEN statements to execute branching, have the program print out the taxes for that employee based upon the following schedule:

10000 - 19999.99	10%
20000 - 29999.99	20%
30000 - 39999.99	30%
40000 - 49999.99	40%

Allow for an income only between 10000 and 49999.99, inclusive.

```
RUN
SALARY? 40000
40000 16000
```

```
READY.
```

# Chapter 5 Test Solutions

The credit for each problem is given in brackets [ ].

Part I [ 5@ = 20 ]

- |    |                 |    |                                   |             |             |
|----|-----------------|----|-----------------------------------|-------------|-------------|
| a) | RUN<br>ABCBC    | c) | RUN<br>MMOMY                      |             |             |
| b) | RUN<br>D<br>END | d) | RUN<br>2<br>4<br>6<br>BREAK IN 30 | 1<br>2<br>3 | 3<br>3<br>3 |

Part II [ 5@ = 20 ]

- Line 20: J is used as the loop variable so it shouldn't be in the input list.
- Line 50: A value for K will create a SYNTAX ERROR during the 3rd iteration of the FOR...NEXT loop.
- There is no line 100 to GOSUB to. Also, 55 END is needed.
- No errors.

Part III

- ```

10 PRINT "A","B","EQUAL"
20 FOR X = -10 TO 10
30   FOR Y = -10 TO 10
40     IF Y = 0 THEN STOP
50     IF X = 0 THEN STOP
60     IF (X + Y) = 0 THEN STOP
70     A = (X / Y) + (Y / X) + (1 / (X + Y))
80     B = (X * Y) ^ 2
90     PRINT A,B,
100    IF A = B THEN PRINT "Y":Z = Z + 1
110   NEXT Y
120  NEXT X
130 IF Z = 0 THEN PRINT "NO SOLUTIONS FOUND"
140 END

```

- ```

10 INPUT "SALARY";S
20 T = INT(S/10000)
30 IF T < 1 OR T > 4 THEN 10
40 ON T GOSUB 70,80,90,100
50 PRINT S,X
60 END
70 X = S * .1: RETURN
80 X = S * .2: RETURN
90 X = S * .3: RETURN
100 X = S * .4: RETURN

```

[ 30 ]



# Chapter 1-5 Examination

## Part I Multiple Choice

Determine the best response and place the corresponding letter in the blank.

- \_\_\_ 1) Consider the following program:

```
10 READ A$
20 DATA "CALLISTO"
30 PRINT A$
40 DATA "EUROPA", "IO, GANYMEDE"
50 GOTO 10
```

How many times will the computer execute line 30 before an error message is printed?

- a) 1  
b) 2  
c) 3  
d) 4  
e) 5
- \_\_\_ 2) If the command  

```
PRINT "LIST"
```

is executed, the computer will:  
a) display the word LIST on the screen.  
b) print a list of the program currently in memory.  
c) print an error message.  
d) do nothing until "RUN" is entered, then print the word LIST.  
e) do nothing until "RUN" is entered, then print a list of the program currently in memory.
- \_\_\_ 3) Which of the following statements is permissible on the Commodore?  
a) IF X = 5 OR 6 THEN PRINT X  
b) IF X = 5 OR IF Y = 7 THEN PRINT X,Y  
c) IF Y = 17 GOTO 100  
d) IF X = Y = Z THEN PRINT "TRUE"  
e) None of the above statements are permissible.
- \_\_\_ 4) The major difference between GOSUB and the GOTO statement is that GOSUB permits use of  
a) RESUME  
b) another GOTO  
c) another GOSUB  
d) RETURN  
e) ONERR

\_\_\_ 5) Determine the output of the following sequence:

```
10 FOR L = 1 TO 5
20   IF L = 3 THEN STOP
30   I = I + 1
40 NEXT L
RUN
BREAK IN 20
PRINT I
```

- a) 2            b) 3            c) 4            d) 5            e) 6

\_\_\_ 6) Which of the following choices prints one random integer between 2 and 11, inclusive?

- a) PRINT INT ( RND (2) \* 11)  
b) PRINT INT ( RND (0) \* 10) + 2  
c) PRINT INT ( RND (0) \* 9) + 2  
d) PRINT INT ( RND (0) \* 10) \* 2  
e) PRINT INT ( RND (0) + 9) \* 2

\_\_\_ 7) The statement INT (X / 100 + .5) \* 100 ...

- a) Rounds X to the nearest integer.  
b) Rounds X to the nearest tenth.  
c) Rounds X to the nearest decade.  
d) Rounds X to the nearest hundredth.  
e) Rounds X to the nearest hundred.

\_\_\_ 8) Which of the following immediate mode listings does not return the value zero (0)? (all listings are preceded by NEW)

- a) PRINT X  
b) X = 1: PRINT POS(0)  
c) X = 1: PRINT X \* Y  
d) CLR : PRINT X  
e) CLR : PRINT TI

\_\_\_ 9) Taking into consideration the computer's rounding errors, which of the following IF statements will print a result?

- a) IF 3↑3 = 3\*3\*3 THEN PRINT "3↑3"  
b) IF 3↑2 = 3\*3\*3 THEN PRINT "3↑2"  
c) IF 2↑3 = 2\*2\*2 THEN PRINT "2↑3"  
d) IF 5↑2 = 5\*5 THEN PRINT "5↑2"  
e) IF 7↑3 = 7\*7\*7 THEN PRINT "7↑3"

\_\_\_ 10) Based on the schematic array at right, which of the following is a true statement?

- a) A(1,1) = 7  
b) A(1,3) = 7  
c) A(3,1) = 7  
d) A(2,4) = 7  
e) A(4,2) = 7

7	0	5	0	0
0	0	7	3	0
6	4	0	0	2
0	8	0	9	4
0	0	0	4	0

- \_\_\_ 11) Determine the output of the following program:
- ```

10 FOR X = 1 TO 3
20   FOR Y = 1 TO 4
30   NEXT Y
40 NEXT X
50 PRINT X;"↑";Y

```
- a) 3 ↑ 4   b) 4 ↑ 3   c) 4 ↑ 4   d) 5 ↑ 4   e) 4 ↑ 5
- \_\_\_ 12) Which of the following statements will erase the current program in memory?
- a) CLR  
b) RESTORE  
c) GET  
d) POS  
e) NEW
- \_\_\_ 13) Four of the following expressions have the same value. Which one is different?
- a)  $3 \uparrow 2 - 1$   
b)  $6 * 2 + 4 / 2$   
c)  $2 + 2 * 3$   
d)  $4 + 2 + 6 / 3$   
e)  $2 \uparrow 3$
- \_\_\_ 14) In four of the cases below, if the first command is used in a program, the second is almost always used also. In which case does the second command have no relation to the first.
- a) FOR...NEXT                      d) INPUT...RESTORE  
b) READ...DATA                    e) STOP...CONT  
c) GOSUB...RETURN
- \_\_\_ 15) You suspect a programming error in a program of which part is shown below:
- ```

...
140 Z = R ↑ 2
150 X = X * H
160 Y = Z + X
...

```
- You would like to determine the value of X before it is multiplied by H in line 150. Which of the following diagnostic techniques should you use?
- a) RESTORE  
b) POS(0)  
c) Run the program and press RUN / STOP after line 140 executes. Then type PRINT X in immediate mode.  
d) Run the program and press CTRL-C just after line 140 executes. Then type PRINT X in immediate mode.  
e) Add line 145 STOP, run the program, and then type PRINT X in immediate mode.

Part II Exact Output

Determine the exact output of each of the following programs:

- 1) NEW  
10 PRINT "RAIN"  
20 PRINT "SNOW"  
NEW  
10 PRINT "SLEET"  
30 PRINT "FOG"  
10 PRINT "HAIL"  
RUN
  
- 2) 10 FOR I = 1 TO 3  
20 READ X,X\$,X(I)  
30 NEXT I  
40 PRINT X\$  
50 PRINT X  
60 PRINT X(2)  
70 DATA 10,ACE,3,2,JACK,8,17,QUEEN,4
  
- 3) 10 X = 1  
20 FOR I = 4 TO 8 STEP 2  
30 IF  $I^2 < 40$  AND  $(I - 8) > (-1 * I)$  THEN PRINT I;  
40 X = X + Y  
50 Y = X \* 2  
60 NEXT I  
70 PRINT X,Y



Part III Locate the errors

Circle the programming error(s) within each program given. Explain what must be done to correct the error. If no error exists, state so.

- 1)    10    REM FIND THE SUM OF 2 NUMBERS  
      20    INPUT NUM1  
      30    INPUT NUM2  
      40    TOTAL = N1 + N2  
      50    PRINT "THE SUM IS ;TOTAL"  
      60    END
  
- 2)    10    REM PRINT THE VARIOUS RANDOM NUMBERS  
      20    FOR I = 1 TO 4  
      30    READ L  
      40    PRINT "A RANDOM NUMBER BETWEEN 0 AND";L;"IS";  
          INT ( RND (0) \* L) + 1  
      50    NEXT L  
      60    DATA 2,25,100,1000  
      70    END
  
- 3)    10    TELL WORD FOR INTENDED NUMBER FROM 1 TO 4  
      20    INPUT "ENTER A NUMBER FROM 1 TO 4";N  
      30    ON N GOTO 50,60,70,80  
      40    IF N<1 OR N>4 THEN 20  
      50    PRINT "ONE"  
      60    PRINT "TWO"  
      70    PRINT "THREE"  
      80    PRINT "FOUR"  
      90    END

## Part IV Programs

Write each of the following programs as concisely as possible. Good programming style is important.

- 1) Write a program which will print a list of 20 random odd integers between 10 and 100, inclusive. Do not print any even numbers and do not print any number more than once.
  
- 2) Write a program based on the following outline:
  - a) Have the user enter an integer from 1 to 12 representing his or her month of birth.
  - b) Have the user enter an integer from 1 to 31 representing the day he or she was born.
  - c) Subtract the smaller from the larger (If alike, result is 0).
  - d) Divide the result of c by 2, add 1, and round off the result to the nearest integer.
  - e) If the result is not between 1 and 5, inclusive, repeat step d.
  - f) Print one of the following five 'pre-divined' fortunes based on the result in step e.
    - i. You will inherit a fortune!
    - ii. You will fall in love!
    - iii. You will find much happiness!
    - iv. You will make a great discovery!
    - v. You will receive a 90 on this test!



Part IV

[ 20 ]

```
1) 10 DIM K(20)
    20 FOR I = 1 TO 20
    30 K(I) = INT(RND(0) * 91 + 11)
    40 IF K(I) / 2 = INT(K(I) / 2) THEN 30
    50 Q = 1
    60 IF I = 1 THEN 90
    70 IF K(I) = K(Q) THEN 30
    80 IF Q <> I - 1 THEN Q = Q + 1: GOTO 70
    90 NEXT I
    100 FOR T = 1 TO 20
    110 PRINT K(T)
    120 NEXT T
    130 END
```

[ 20 ]

```
2) 10 INPUT "ENTER MONTH OF BIRTH";M
    20 INPUT "ENTER DAY OF BIRTH";D
    30 IF D > M THEN C = D - M: GOTO 50
    40 C = M - D
    50 C = INT(C / 2 + 1.5)
    60 IF C > 5 OR C < 1 THEN 50
    70 FOR G = 1 TO C
    80 READ A$
    90 NEXT G
    100 PRINT A$
    110 DATA "YOU WILL INHERIT A FORTUNE!"
    120 DATA "YOU WILL FALL IN LOVE!"
    130 DATA "YOU WILL FIND MUCH HAPPINESS!"
    140 DATA "YOU WILL MAKE A GREAT DISCOVERY!"
    150 DATA "YOU WILL RECIEVE A 90 ON THIS TEST!"
    160 END
```

# Lesson 6.1

OBJECTIVES: a) to apply the Commodore-64's ability to display a multitude of shapes and colors

b) to apply the use of cursor control movement for both editing programs and screen control

---

ASSIGNMENT: Read pages 6.1 through 6.8 Do review problems 1 - 2  
Text problems 1,7,11 Homework problems 2,8,10

---

The Commodore has the ability to print text in a variety of colors which can be altered immediately by depressing the CTRL or (Commodore) key along with a number key, ranging from 1 to 8. However, you must emphasize that if the cursor is within quotation marks, such as in a PRINT or assignment statement, a unique graphic code will be displayed instead. The cursor movement keys and the reverse on and off keys will also generate unique graphic symbols when used within quotation marks.

---

As programs become more elaborate and as the quantity of programs on a disk increases, it becomes more and more difficult to recall what each program did (or was supposed to do). Internal documentation with REM statements is useful and important. A second technique, valuable during program execution, is the use of a program 'header'. The header should announce to the program user the name of the program, the author, the date the program was written and the date it was last revised.

Have students suggest a header design. Lay out the header using a grid. Be liberal with blank lines and center the information on the screen. A sample is given on page 6.2. The program used to produce the header shown on that page is as follows:

```
10 PRINT " ";
20 H$="HEADER DEMONSTRATION":L=20
30 PRINT:PRINT:PRINT
40 FOR I=1 TO (21-INT(L/2))
50 PRINT " ";
60 NEXT I
70 PRINT H$
80 REM
90 H$="WRITTEN BY: JOHN WANG":L=21
100 PRINT:PRINT
110 FOR I=1 TO (21-INT(L/2))
120 PRINT " ";
130 NEXT I
140 PRINT H$
150 REM
160 D1$="7/6/84"
170 D2$="7/24/84"
180 PRINT:PRINT:PRINT:PRINT:PRINT:PRINT
190 PRINT " DATE OF CREATION: ";D1$
200 PRINT " DATE OF LAST REV: ";D2$
```

Notice the variable L in lines 20 and 70. This variable represents the string length (in characters) of the program name and then the author's name. (Re-using variables is an effective practice.) As the program is revised, line 130 is easily changed. The entire segment can be used at the beginning of any program. The programmer need not retype the entire segment. Instead, only lines 20, 120 and 130 need to be modified.

```
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33
HEADER DEMONSTRATION
WRITTEN BY: JOHN WANG
DATE OF CREATION: 7/6/84
DATE OF LAST REV: 7/24/84
```

# Worksheet 6.1

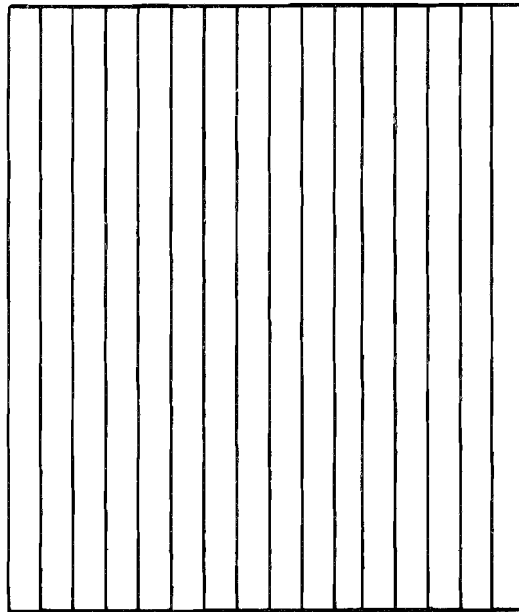
Part I Determine the exact output of each of the following programs:

- a) 

```
10 FOR I = 1 TO 1000:PRINT " #";:NEXT I
```
- b) 

```
10 FOR I = 1 TO 10
20   FOR J = 1 TO I
30     PRINT "X";
40   NEXT J
50   PRINT
60 NEXT I
```

Part II Write a program to print all 16 colors in vertical columns as shown below using only one print statement:







## Lesson 6.2

OBJECTIVES: a) to apply PEEK and POKE commands for screen control, lowercase mode, foreground color and screen memory  
b) to apply the joystick in programs

---

ASSIGNMENT: Read pages 6.9 through 6.25 Do review 3 - 6  
Text problems 3,5,13 Homework problems 6,12

---

PEEK and POKE are functions that allow the user to see what a value a particular byte of memory contains and to change the value of a byte in memory. The following program introduces the use of PEEK and POKE:

```
10 POKE 52380,INT(15*RND(0)):REM BORDER
20 POKE 53281,INT(15*RND(0)):REM BACKGROUND
30 GET A$:IF A$ = "" THEN 30
40 IF A$<>"E" THEN 10
```

This program will allow the user to choose a suitable border and background color by continually displaying random combinations until the key 'E' is pressed. The statements in lines 10 and 20 POKE random values between 0 and 15, inclusive, into memory locations that determine the color for the border and the background, respectively.

You may decide not to cover the material in the text on joysticks. If you do wish to, the discussion in the text should be adequate.



# Worksheet 6.2

Part I Describe what each of the following programs would do if they were run:

a) 

```
10 PRINT " ";
20 FOR I=1 TO 1000
30 PRINT " ";:REM MOVE CURSOR UP
40 NEXT I
```

b) 

```
10 FOR I=1 TO 50
20 PRINT
30 NEXT I
40 PRINT " ";
```

Part II Write each of the following subroutines as concisely as possible:

a) Often in a program you will need to tab to a certain position on the screen, so write a subroutine where the variables X and Y are the coordinates. (Use only cursor movement keys.)

b) Because the above subroutine is quite slow, especially if you are tabbing to (39,24), you will need to think of a faster routine. Examine the table on page 6.10 and rewrite the program using that information. (Hint: PEEKs and POKEs)

# Worksheet 6.2 Solutions

## Part I

- a) This program will produce the same output even if line 20 were omitted.
- b) This program clears the screen and moves the cursor to home.

## Part II

- a)

```
10 PRINT "A":
20 IF X=0 THEN GOTO 50
30 FOR I=1 TO X
40 PRINT "A":
50 NEXT I
60 IF X=0 THEN RETURN
70 FOR J=1 TO X
80 PRINT "A":
90 NEXT J
100 RETURN
```
  
- b)

```
10 FOR E=211.4:PRINT 214.4
20 FOR E=201.4+(1024+40*Y)/60000
30 FOR E=210:INT((1024+40*Y)/256)
40 RETURN
```

# Lesson 6.3

OBJECTIVE: a) to apply the sound capabilities of the Commodore-64

---

ASSIGNMENT: Read pages 6.26 through 6.34  
Text problem 9

Do review 7 - 8  
Homework problem 4

---

When writing programs that use PEEKs and POKEs, you will find yourself continually referring to a chart for memory locations. One way to reduce the use of PEEKs and POKEs and to minimize silly errors that can occur is to incorporate POKEs and PEEKs into a subroutine, as below:

```
10000 REM A(1)...TONE
10010 REM A(2)...ATTACK
10020 REM A(3)...DECAY
10030 REM A(4)...SUSTAIN
10040 REM A(5)...RELEASE
10050 REM A(6)...PEAK VOLUME
10060 REM A(7)...SOUND CONTROL REGISTER
10100 POKE 54273,A(1)+INT(A(1)/256)*256:POKE 54273,INT(A(1)/256):PCH TONE
10110 POKE 54277,A(2)*16+A(3):REM ATTACK AND DECAY
10120 POKE 54278,A(4)*16+A(5):REM SUSTAIN AND RELEASE
10130 POKE 54276,A(6):REM VOLUME
10140 RETURN
11000 CALL 54276,A(7)
11010 RETURN
12000 POKE 51276,A(7) AND 254
12010 RETURN
```

The above program stores the values to be POKEd in an array which can then be POKEd by a simple GOSUB statement. Such a technique will reduce errors and make a program much easier to follow.



# Worksheet 6.3

Carefully analyze the following program which turns the computer keyboard into an organ and then write a line by line description of what each line does.

```
10 DIM S$(40),S(40):PRINT "M":K=34
20 FOR I=1 TO 7
30   READ A(I)
40 NEXT I
50 FOR I=1 TO K
60   READ S$(I),S(I)
70 NEXT I
80 GET A$:IF A$="" THEN 80
90 GOSUB 12000
100 FOR I=1 TO K
110 IF A$=S$(I) THEN 140
120 NEXT I
130 GOTO 80
140 A(1)=S(I)
150 GOSUB 10000
160 GOSUB 11000
170 GOTO 80
1000 DATA 10000,0,15,0,0,15,33
1010 DATA Z,2145,S,2273,X,2408,D,2551,C,2703,V,2864,G,3034
1020 DATA B,3215,H,3406,N,3608,J,3823,M,4050,"",",4291
1030 DATA L,4547,".",",4817,":",5103,"/",5407
1040 DATA Q,4291,2,4547,W,4817,3,5103,E,5407,R,5728,S,6069,
      T,6430,6,6812,Y,7217
1050 DATA 7,7647,U,8101,I,8583,9,9094,O,9634,0,9634,F,10207
9999 END
10000 REM
10010 REM POKE ARRAY INTO MEMORY LOC.
10020 REM
10030 REM A(1)...TONE
10040 REM A(2)...ATTACK
10050 REM A(3)...DECAY
10060 REM A(4)...SUSTAIN
10070 REM A(5)...RELEASE
10080 REM A(6)...PEAK VOLUME
10090 REM A(7)...SOND CONTROL REGISTER
10100 POKE 54272,A(1)-INT(A(1)/256)*256:POKE 54273,INT(A(1)
      /256):REM TONE
10110 POKE 54277,A(2)*16+A(3):REM ATTACK AND DECAY
10120 POKE 54278,A(4)*16+A(5):REM SUSTAIN AND RELEASE
10130 POKE 54296,A(6):REM VOLUME
10140 RETURN
11000 REM
11010 REM TURN ON SOUND
11020 REM
11030 POKE 54276,A(7)
11040 RETURN
12000 REM
12010 REM TURN OFF SOUND
12020 REM
12030 POKE 54276,A(7) AND 254
12040 RETURN
```

# Worksheet 6.3 Solutions

LINE 10: The array S\$() will contain characters that produce the corresponding tone, S(). The print statement clears the screen, and variable K is the number of keys defined for each separate note.

LINE 20: Begin loop.

LINE 30: Read in Tone, Attack, Decay...

LINE 40: End loop.

LINE 50: Begin loop.

LINE 60: Read in keys and corresponding frequencies.

LINE 70: End loop.

LINE 80: Wait for input.

LINE 90: Turn off sound.

LINE 100: Begin loop.

LINE 110: Check whether the key entered is one of the notes; if so, then goto line 140.

LINE 120: End loop.

LINE 130: Goto main loop.

LINE 140: Assign the new frequency to the one corresponding to the key entered.

LINE 150: POKE new values into the Tone, Attack, Decay... memory locations.

LINE 160: Turn on sound.

LINE 170: Go back to main loop at line 80.

LINE 1000: Data statement for Tone, Attack, Decay...

LINE 1010-1050: Data statements for key and frequency definitions.

LINE 10000-10140: This subroutine POKES the values of Tone, Attack, Decay, Sustain, Release and Peak values into the array into their corresponding memory locations.

LINE 11000-11040: Jumping to this subroutine will play note POKED by the subroutine beginning at 10000.

LINE 12000-12040: Turn off sound initiated by subroutine beginning at 11000.



# Chapter 6 Supplementary Problems

- 1) Write an etch-a-sketch program where the user manipulates the joystick to move the cursor in any of 8 different directions, plotting an asterisk in foreground color if the joystick button is pressed.
- 2) Write a program employing the sawtooth wave that acts as a siren. Allow the frequency to "slide" back and forth between the C and D in the 6th octave. Consult the chart on page 6.27 for tone codes. Be sure to set the ADSR envelope to allow the sound to be heard.

# Chapter 6 Supplementary Problem Solutions

- 1) 10 X=0:Y=0:PRINT " ";  
20 B=15-PEEK(56321) AND 15):IF J=0 THEN 1000  
30 ON J GOSUB 100,200,0,300,400,500,0,600,700,800  
40 IF X>39 THEN X=39  
50 IF X<0 THEN X=0  
60 IF Y>24 THEN Y=24  
70 IF Y<0 THEN Y=0  
80 POKE211,X:POKE214,Y:POKE209,((1024+40\*Y) AND 255):POKE210,INT(  
(1024+40\*Y)/256)  
90 IF B=0 THEN PRINT " ";  
99 GOTO 20  
100 Y=Y-1:RETURN  
200 Y=Y+1:RETURN  
300 X=X-1:RETURN  
400 X=X-1:Y=Y-1:RETURN  
500 X=X-1:Y=Y+1:RETURN  
600 X=X+1:RETURN  
700 X=X+1:Y=Y-1:RETURN  
800 X=X+1:Y=Y+1:RETURN  
1000 GET A\$:IF A\$<>"Q" THEN 20
- 2) 10 X=54272  
20 A=5  
30 D=5  
40 POKE X+5,A\*16+D  
50 S=10  
60 R=7  
70 POKE X+6,S\*16+R  
80 POKE X+24,15  
90 A=8583:B=9634  
100 A1=A:B1=B  
110 POKE X+4,33  
120 Q=Q\*10  
130 FOR W=A TO B STEP Q  
140 POKE (X+1),INT(W/256)  
150 POKE X,W-INT(W/256)\*256  
160 NEXT W  
170 IF A=A1 THEN B=A1:A=B1:Q=-1:GOTO 120  
180 A=A1:B=B1:Q=1:GOTO 120

# Chapter 6 Test

Part I Determine the exact output of each of the following programs:

- a) `10 POKE 53280,0:REM BORDER`  
`20 POKE 53281,0:REM BACKGROUND`  
`30 FOR I=55296 TO 56295:POKE I,0:NEXT I:REM FOREGROUND`
- b) `10 FOR I=1 TO 25`  
`20 PRINT`  
`30 NEXT I`  
`40 PRINT PEEK(211);",";PEEK(214):REM PEEK(211)=X POSITION ON`  
`SCREEN`  
`50 REM PEEK(214)=Y POSITION ON SCREEN`

Part II Circle the programming error(s) within each program given. Explain what must be done in order to correct the error. If no error exists, state so.

- a) `10 POKE (646,1):REM CHANGE FOREGROUND COLOR TO WHITE`
- b) `10 FOR I=0 TO 1000`  
`20 PRINT PEEK(I)`  
`30 NEXT I`
- c) `10 A=255:POKE 123,A+1`
- d) `10 PRINT PEEK 646:REM PRINT FOREGROUND COLOR`

Part III

- a) Write a program that will draw a solid square in foreground color at the middle of the screen. The size of the square is entered by the user and can range from 1 to 15 blocks in length.
- b) Utilizing foreground, border and background, create a light show by randomly changing the border and background color, and by printing a random number of different graphic characters ranging in ASC() values from 64 to 127 in various positions on the screen.

The addresses of the memory locations are given below:

Foreground : 646  
Border : 53280  
Background : 53281

# Chapter 6 Test Solutions

The credit for each problem is given in brackets [ ].

Part I [ 8@ = 16 ]

- a) This program turns the screen to black.
- b) 0 , 24

Part II [ 6@ = 24 ]

- a) The arguments in POKE should not be enclosed in ().
- b) No errors.
- c) An ?ILLEGAL QUANTITY ERROR will exist; only values between 0 and 255, inclusive, may be POKEd into any memory location.
- d) Parentheses are needed to enclose the arguments in PEEK.

Part III

- a) 

```
10 INPUT "LENGTH";L
20 PRINT " ";
30 IF L<1 OR L>15 THEN 10
40 FOR I=1 TO 12-INT(L/2)
50 PRINT
60 NEXT I
70 FOR I=1 TO L
80 FOR J=1 TO 20-INT(L/2)
90 PRINT " ";
100 NEXT J
110 FOR J=1 TO L
120 PRINT " "
130 NEXT J
140 NEXT I
```

 [ 30 ]

- b) 

```
10 PRINT " ";
20 IF RND(0)<.5 THEN POKE 53280,INT(16*RND(0))
30 IF RND(0)<.5 THEN POKE 53281,INT(16*RND(0))
40 FOR I=5 TO INT(10*RND(0))
50 X=INT(40*RND(0))
60 Y=INT(25*RND(0))
70 POKE 1024+40*Y+X,INT(64*RND(0))+64
80 POKE 55296+40*Y+X,INT(16*RND(0))
90 NEXT I
100 GET A$:IF A$<>"Q" THEN 20
```

 [ 30 ]

# Lesson 7.1

OBJECTIVE: a) to apply the functions SQR, ABS and SGN

---

ASSIGNMENT: Read pages 7.1 and 7.2 Do review problems 1 and 2  
Text problems 1,5,15,19 Homework problems 2,4,6,16

---

This chapter should be covered only in courses where an emphasis is being placed on mathematical applications. Much of the material in both the text and teacher's guide will be inappropriate for use with students who have not had at least an introduction to trigonometry.

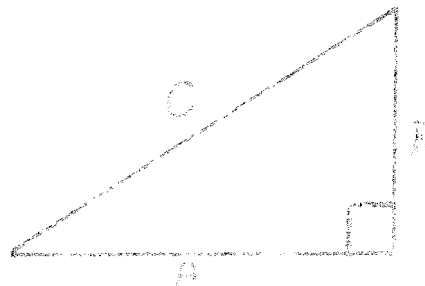
Part I of Worksheet 7.1 for this lesson will give students some practice with the simple functions SQR, ABS and SGN.

Taking the square root of  $X$  is equivalent to raising  $X$  to the .5 power. Similarly then, the cube root may be found by raising  $X$  to the one third power.

---

You may find the following applications interesting when you introduce the material in this lesson.

The Pythagorean Theorem gives this relationship between the lengths of the sides of any right triangle:



$$C^2 = A^2 + B^2$$

$$C = \sqrt{A^2 + B^2}$$

Now, a computer program involving this relationship, must use the SQR function.

In a science lab, measurements are often made using trials and a result is obtained from the mean (average) value of the measurements. But consider the following two sets of data:

STUDENT 1	STUDENT 2
6.32	6.41
6.35	6.36
6.37	6.37
6.39	6.45
6.37	6.31

The mean is 3.36 for both sets, but student 1 has clearly more consistent measurements. This student's results are said to have greater precision and less uncertainty. The uncertainty can be quantified by determining, on the average, how much each measurement differs from the mean. Such a difference is called a deviation, and the sign of the deviation is irrelevant. The deviation for a particular measurement is thus given by:

$$| \text{MEAN} - \text{MEASUREMENT} |$$

The following program demonstrates the use of ABS for determining the average deviation of a set of data:

```
10 REM INPUT DATA, FIND MEAN
20 REM
30 FOR I = 1 TO 5
40 INPUT "ENTER DATA POINT";X(I)
50 T = T + X(I)
60 NEXT I
70 M = T / 5
80 REM
90 REM CALCULATE DEVIATIONS,
100 REM FIND AVERAGE DEVIATION
110 REM
120 FOR I = 1 TO 5
130 D(I) = ABS(M-X(I))
140 S = S + D(I)
150 NEXT I
160 REM
170 REM PRINT RESULTS
180 REM
190 PRINT: PRINT"DATA PTS. ","DEVIATION"
200 FOR I = 1 TO 5
210 PRINT X(I), INT(D(I) * 1000 + .5) / 1000
220 NEXT I
230 PRINT
240 PRINT"MEAN ="; INT(M * 1000 + .5) / 1000
250 S = S / 5
260 PRINT "UNCERTAINTY = +OR-"; INT(S * 1000 + .5) / 1000
270 END
```

Numbers have been rounded to the nearest one-thousandth in lines 210,240 and 260. When this program is run, student 1 would report a measurement of  $6.36 \pm .02$  while student 2 would report  $6.36 \pm .06$ :

# Worksheet 7.1

Part I Determine the output of each of the following programs:

- a) 

```
10 FOR I = 1 TO 6
20   READ X
30   PRINT SQR(X)
40 NEXT I
50 DATA 16,9,4,1,0,-1
```
- b) 

```
10 FOR I = 1 TO 3
20   READ X
30   PRINT SQR ( SQR (X))
40   NEXT I
50 DATA 1,16,81
```
- c) 

```
10 FOR I = -2 TO 2
20   PRINT I; TAB( 4); ABS (I ↑ 3); TAB( 7); SGN (I ↑ 3)
30 NEXT I
```
- d) 

```
10 FOR I = -2 TO 2
20   PRINT SGN (I) - ABS (I)
30 NEXT I
```

Part II Write each of the following programs as concisely as possible:

- a) Write a program to find the values of the function  $-X - X^2 + 6$  where  $f(X)$  is positive and  $X$  is an integer between  $-10$  and  $10$ .
- b) Find the  $X$  values where both  $X - 5$  and  $-X + 10$  are positive.

# Worksheet 7.1 Solutions

Part I

a) 4  
3  
2  
1  
0

c) -2 8 -1  
-1 1 -1  
0 0 0  
1 1 1  
2 8 1

ILLEGAL QUANTITY ERROR IN 30

b) 1  
2  
3

d) -3  
-2  
0  
0  
-1

Part II

```
1) 10 FOR X = -10 TO 10
    20 IF  $-(X + 2) - X + 6 > 0$  THEN PRINT X
    30 NEXT X
```

RUN

6  
-1  
0  
1

```
2) 10 FOR X = -10 TO 10
    20 IF  $X - 5 > 0$  AND  $-X + 10 > 0$  THEN PRINT X
    30 NEXT X
```

RUN

5  
6  
7  
8  
9

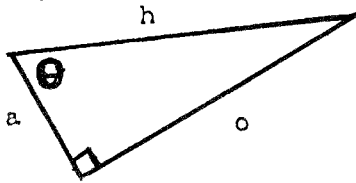


# Lesson 7.2

- OBJECTIVES:
- a) to understand the relationship between degree and radian measure
  - b) to gain experience with the SIN and COS functions
  - c) to apply the trigonometric functions SIN, COS and TAN to right triangle problems
  - d) to learn to simulate SIN<sup>-1</sup> and COS<sup>-1</sup> using the ATN function, i.e., arcsine and arcosine

ASSIGNMENT:    Read pages 7.3 through 7.5    No review problems  
                   Text problems 3,7,21,23,25    Homework problems 8,10,26

Your students may need to be shown how to apply the trigonometric functions to the right triangles. The following figure summarizes the right triangle relationships:

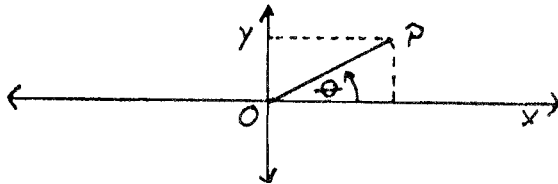


$$\text{SIN } \theta = \frac{\text{opposite side}}{\text{hypotenuse}} = \frac{o}{h}$$

$$\text{COS } \theta = \frac{\text{adjacent side}}{\text{hypotenuse}} = \frac{a}{h}$$

$$\text{TAN } \theta = \frac{\text{opposite side}}{\text{adjacent side}} = \frac{o}{a}$$

For problems on a coordinate system the same relationships hold except that theta ( $\theta$ ) is often taken to be the angle between ray OP and the positive x-axis, as indicated in the following diagram:



For the coordinate system shown above, the tangent relation is then

$$\text{TAN } \theta = \frac{Y}{X}$$

If the angle is the unknown, it may be found by taking the 'inverse tangent' of both sides of the above equation, giving

$$\theta = \text{ATN } \frac{Y}{X}$$

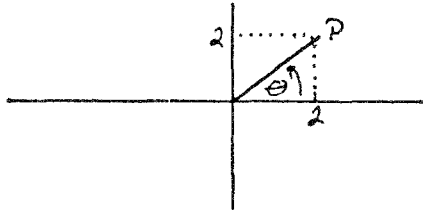
The following program will determine theta when X and Y are known:

```

10 C = 180 / 3.14159265
20 INPUT "X";X
30 INPUT "Y";Y
40 A = C * ATN (Y / X)
50 PRINT "THETA IS";A;"DEGREES"

```

Consider the following example where point P lies in the first quadrant:

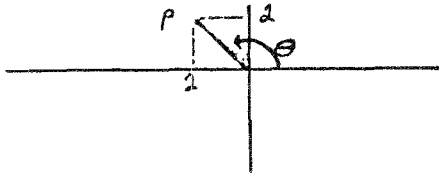


```

RUN
X? 2
Y? 2
THETA IS 45.0000001 DEGREES

```

But notice the result when the point lies in the second quadrant:



```

RUN
X? -2
Y? 2
THETA IS -45.0000001 DEGREES

```

Similar problems exist in the third and fourth quadrants. The problem may be modified as follows:

```

10 C = 180 / 3.14159265
20 INPUT "X";X
30 INPUT "Y";Y
40 A = C * ATN (Y / X)
50 REM
60 IF X < 0 THEN A = A + 180
70 IF X > 0 AND Y < 0 THEN A = A + 360
80 PRINT "THETA IS";A;"DEGREES"

```

For the second quadrant example above, the result is now

```

RUN
X? -2
Y? 2
THETA IS 135 DEGREES

```

Third quadrant:

```

RUN
X? -2
Y? -2
THETA IS 225 DEGREES

```

Fourth quadrant:

```

RUN
X? 2
Y? -2
THETA IS 315 DEGREES

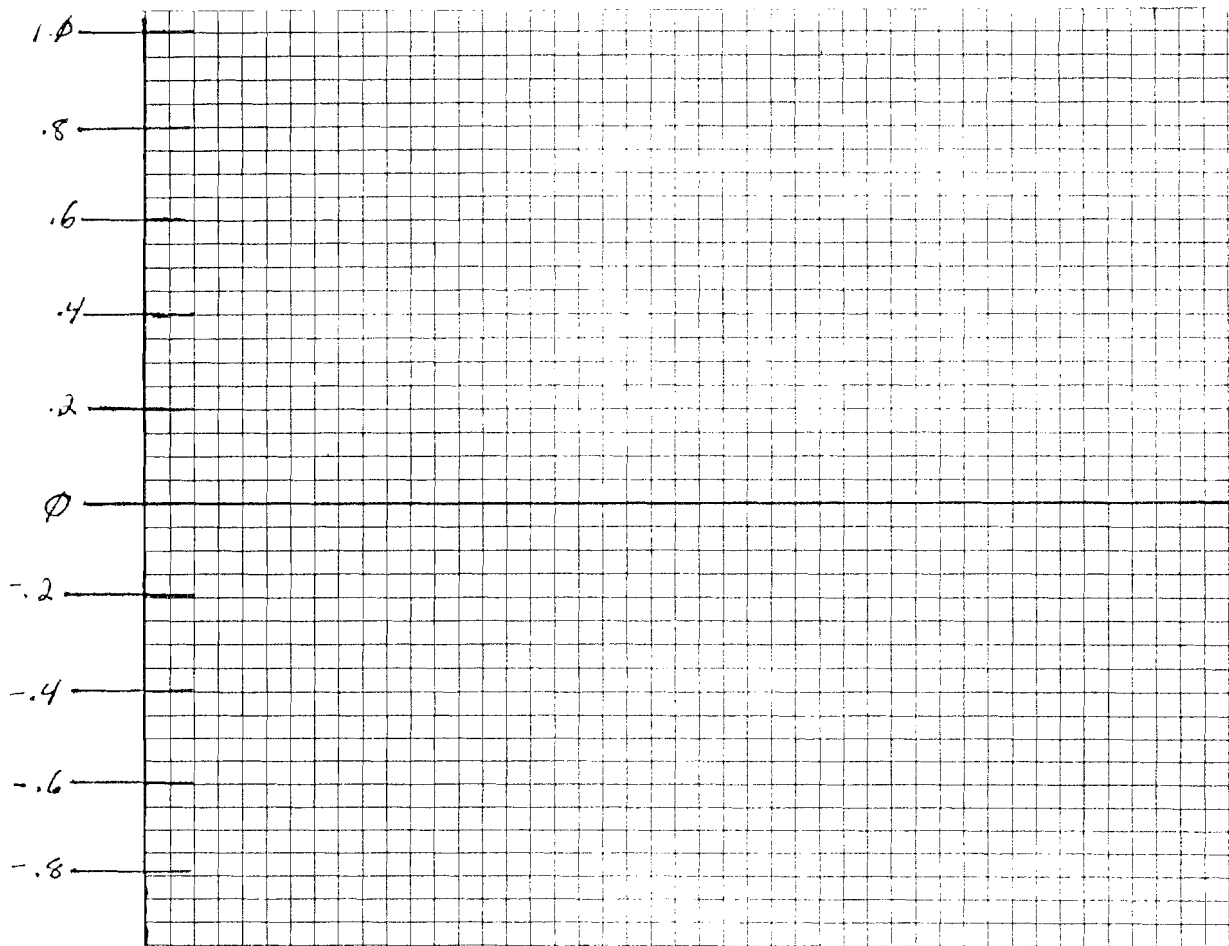
```

# Worksheet 7.2

Part I Use the data given in the table to sketch the graph of  $\text{SIN}(X)$ . Note that you must convert from radians to degrees:

X (rad)	SIN(X)	X (rad)	SIN(X)
0	0	$9\pi/8$	-.38
$\pi/8$	.38	$5\pi/4$	-.71
$\pi/4$	.71	$11\pi/8$	-.92
$3\pi/8$	.92	$3\pi/2$	-1
$\pi/2$	1	$13\pi/8$	-.92
$5\pi/8$	.92	$7\pi/4$	-.71
$3\pi/4$	.71	$15\pi/8$	-.38
$7\pi/8$	.38	$2\pi$	0
$\pi$	0		

The data is rounded to the nearest hundredth. Make sure that you connect the points with a smooth curve.

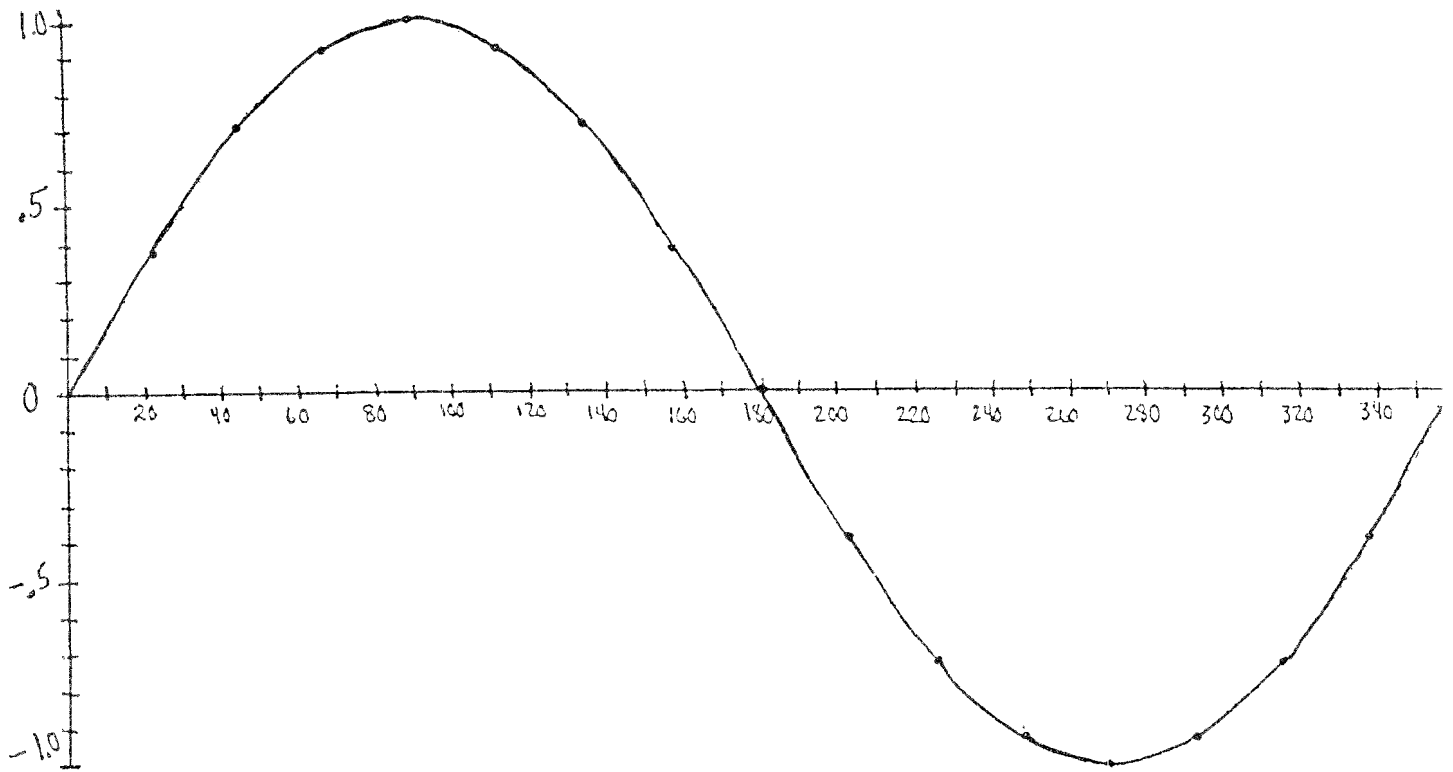


Part II Use the graph above to help you complete the following exercises:

- Estimate the angle whose sine is .5 (i.e., determine the inverse sine of .5).
- A right triangle contains an angle of 15 degrees and has a hypotenuse of length 10. What is the length of the opposite side? The adjacent side?

# Worksheet 7.2 Solutions

## Part I



To plot points on the graph, the student must convert from radians to degrees; each X value is multiplied by  $360/2\pi$ , or  $180/\pi$ . As an example, take the X value  $\pi/4$ :

$$\frac{\pi}{4} \times \frac{180}{\pi} = 45$$

Many students will quickly realize that  $\pi/8$  rad is the difference between successive X values, so all of the points are multiples of 22.5 degrees.

## Part II

a) 30 degrees.

b)  $\text{SIN } \theta = \frac{o}{h}$  and  $\text{COS } \theta = \frac{a}{h}$

So,  $o = h \text{ SIN } \theta = (10) (\text{SIN } 15)$  and  $a = h \text{ COS } \theta = (10) (\text{COS } 15)$

SIN 15 can be read from the graph as approximately .25 or .26, so the length of the opposite side is 2.6. The cosine of 15 degrees is approximately .96, so the length of the adjacent side is about 9.6.

# Lesson 7.3

OBJECTIVES: a) to gain experience with the LOG and EXP functions  
b) to distinguish between a natural log and a base 10 log

---

ASSIGNMENT: Read pages 7.5 and 7.6  
Text problems 9,13 Homework problems 14,18,24

---

An expression is given in the text for finding common logarithms. The common logarithm of a positive number N is the exponent P to which 10 must be raised to produce the number N. So

$$10^P = N$$

and

$$\text{LOG}_{10} N = P$$

The relationship above can be generalized to include any base

$$B^P = N$$

and

$$\text{LOG}_B N = P$$

For example, consider the problem  $2^P = 64$ . The solution is found from the following program:

```
10 INPUT "NUMBER";N
20 INPUT "BASE";B
30 T = LOG (N) / LOG (B)
40 PRINT B;"RAISED TO THE POWER";T;"=";N
```

```
RUN
NUMBER? 64
BASE? 2
2 RAISED TO THE POWER 6 = 64
```

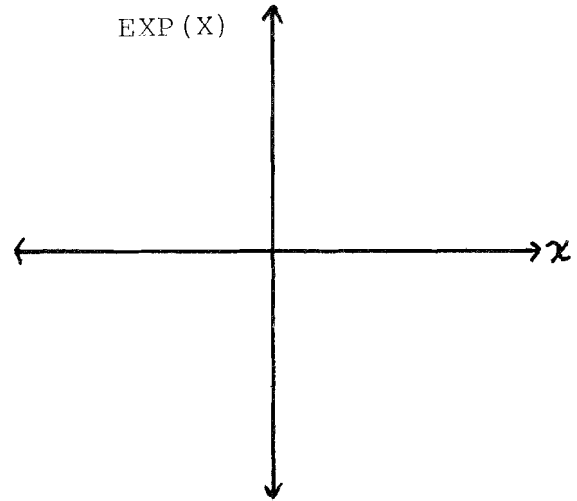


# Worksheet 7.3

For each case given below, use the table of data to plot points on the grid. The data is rounded to the nearest tenth for simplicity. After the points are plotted, connect them with a smooth curve. Note: choose the scales of the axes so that the graph of each function is as large as possible without extending beyond the grid. After you have completed the graph, answer the associated questions:

## Part I Exponential function

X	EXP(X)
-2	.1
-1.5	.2
-1	.4
-.5	.6
0	1
.5	1.6
1	2.7
1.5	4.5
2	7.4

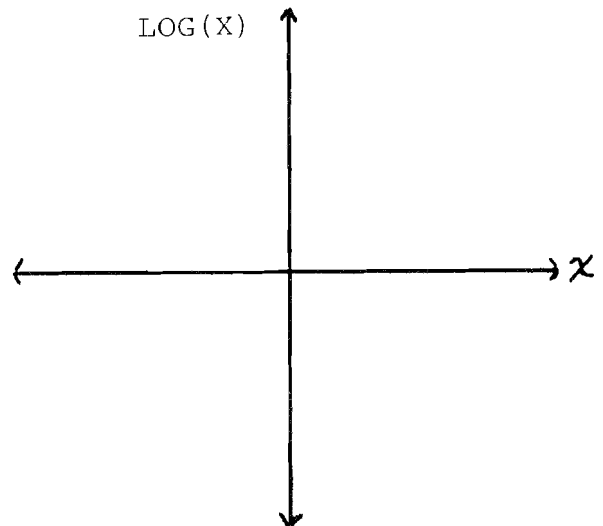


a) At what value of X do you believe EXP(X) will become negative?

b) As the X value increases, to what value does the slope of the curve seem to converge?

## Part II Logarithmic function

X	LOG(X)
.2	-1.6
.4	-.9
.8	-.2
1.2	.2
1.6	.5
2	.7
2.4	.9
3	1.1



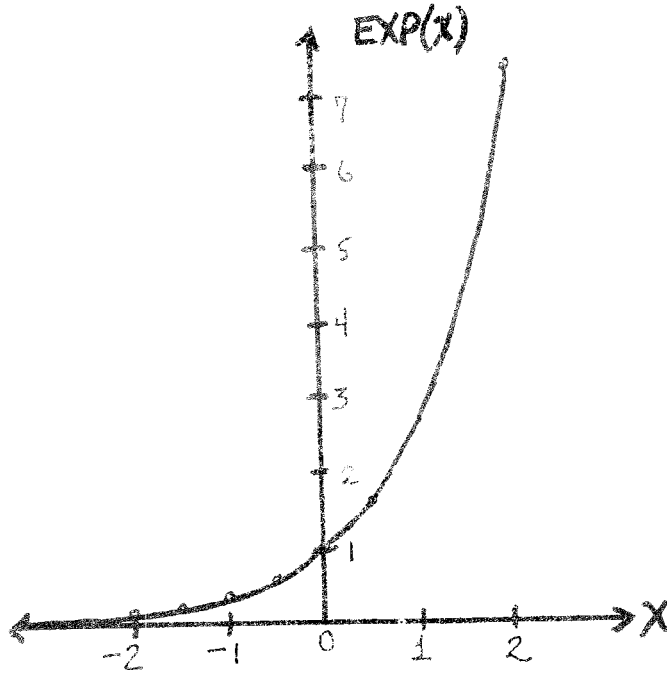
a) What do you believe will happen when the computer tries to perform LOG(-1)?

b) Estimate the value of X which makes LOG(X) = 1.

# Worksheet 7.3 Solutions

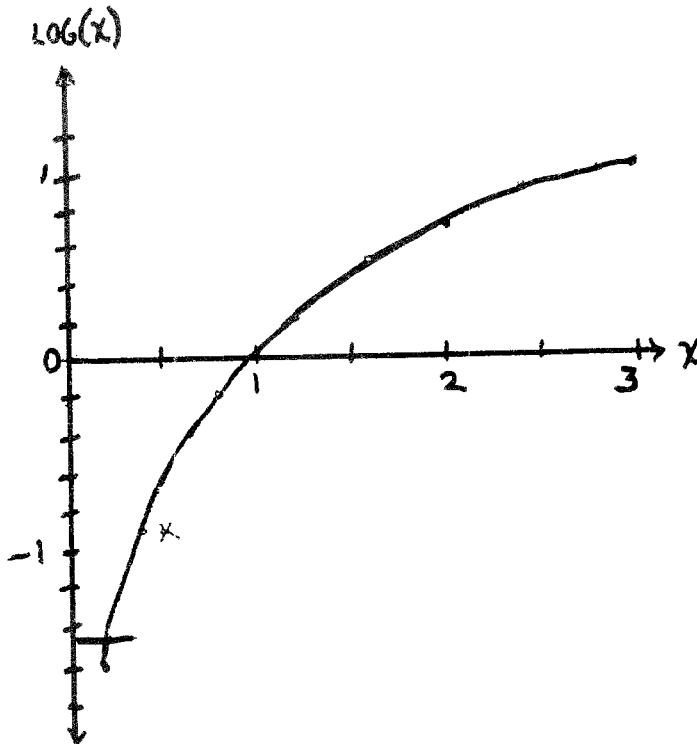
## Part I

- a) It will not become negative.
- b) Infinite



## Part II

- a) ERROR
- b) 2.8 approximately.





# Lesson 7.4

- OBJECTIVES: a) to learn to define functions using DEF  
b) to recognize the advantages of using the DEF statement
- 

ASSIGNMENT: Read pages 7.6 and 7.7 Do review problem 3  
Text problems 11,17 Homework problems 12,20,22

---

User-defined functions are not very powerful because only one argument is allowed per function. Nevertheless, they are convenient, especially when a function will be used a number of times within a program.

Students tend to be confused by the use of a variable name to name the function and the use of a 'dummy' variable for the expression. For example:

```
DEF FNJ(Z) = Z ↑ 3 + Z ↑ 2 + Z
```

J is the function's name and any later reference to the function will be made by the name J. The Z variable is a 'dummy'; it could be any variable. What especially confuses the students is that later on when we evaluate the function, we do not usually use the variable Z but instead substitute whatever variable we want into the function named J. The following program helps clarify this by evaluating two functions named J and M:

```
10 PRINT "X", "FN J(X)", "FN M(X) "  
20 DEF FN J(Z) = Z ↑ 3 + Z ↑ 2 + Z  
30 DEF FN M(S) = 5 * S + 6  
40 FOR X = 1 TO 5  
50 PRINT X, FN J(X), FN M(X)  
60 NEXT X
```

```
RUN  
X          FN J(X)    FN M(X)  
1           3          11  
2          14          16  
3          39          21  
4          84          26  
5         155          31
```

Notice that the variables Z and S in lines 10 and 20 are 'dummies'; any variable name could be used. When the functions are evaluated at line 40, X is substituted for the Z in the function named J, and the X is substituted for S in the function named M.



# Worksheet 7.4

Part I Predict the output of each of the following programs:

- a) 

```
10 DEF FN Q(I) = SQR (I) + I
20 FOR I = 1 TO 3
30 READ J
40 PRINT FN Q(J)
50 NEXT I
60 DATA 9,16,25
```
- b) 

```
10 DEF FN G(X) = (2 + SGN (-X)) / X
20 FOR X = -3 TO 3 STEP 2
30 PRINT FN G(X)
40 NEXT X
```
- c) 

```
10 DEF FN H(X) = ABS (X ^ 2 - 2 * X - 3)
20 PRINT FN H(0)
30 PRINT FN H(4) - FN H(-2)
```
- d) 

```
10 DEF FN X(X) = X ^ 3 - 1
20 FOR I = 1 TO 3
30 PRINT FN X(I)
40 NEXT I
```

Part II Write each of the following programs as concisely as possible:

- a) XYZ Manufacturing has a fixed pay rate of \$4.55 for all employees. If a person works overtime (HOURS>40), then he or she receives time and a half for those hours. Write a program using DEF functions which will figure out the amount of gross pay that an employee will receive. Use only one calculation (function) per employee. Also check the input for invalid data. HINT: you'll need to work with two separate functions.

```
RUN
NAME AND HOURS PLEASE? MIKE,-9
NAME AND HOURS PLEASE? MIKE,37
MIKE      168.35
ANOTHER Y/N?
NAME AND HOURS PLEASE? FRED,52
FRED      263.9
ANOTHER Y/N?
```

- b) Write a program to solve the following polynomial for the values from -3 to 5 in increments of .75. Have your answers rounded to the nearest thousandth.

$$F(X) = X^3 + 3X - \text{SIN}(X)$$

# Worksheet 7.4 Solutions

## Part I

- a) RUN  
12  
20  
30
- b) RUN  
-1  
-3  
1  
.333333333
- c) RUN  
3  
0
- d) RUN  
0  
7  
26

## Part II

- a) 10 DEF FN R(X) = 4.55 \* X  
20 DEF FN J(Z) = (4.55 \* 40) + ((Z - 40) \* (1.5 \* 4.55))  
30 INPUT "NAME AND HOURS PLEASE";N\$,H  
40 IF H < 0 THEN 30  
50 IF H <= 40 THEN P = FN R(H): GOTO 70  
60 P = FN J(H)  
70 PRINT N\$,P  
80 PRINT "ANOTHER Y/N"  
90 GET A\$: IF A\$ = "" THEN 90  
100 IF A\$ = "Y" THEN 30  
110 END
- b) 10 REM TABLE OF A FUNCTION  
20 DEF FN F(X) = X ↑ 3 + 3 \* X - SIN (X)  
30 PRINT "J", "FN J(X)": PRINT "-----", "-----"  
40 FOR J = -3 TO 5 STEP .75  
50 PRINT J, INT (FN F(J) \* 1000 + .5) / 1000  
60 NEXT J

```
RUN
J          F(J)
-----
-3         -35.854
-2.25     -17.358
-1.5      -6.873
-.75      -1.986
0          5E-03
.75       1.995
1.5       6.882
2.25     17.367
3         35.863
3.75     64.56
4.5     105.607
```

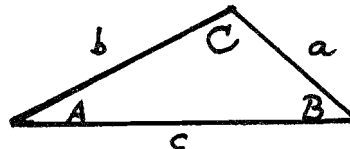
# Chapter 7 Supplementary Problems

1) Write a program and compare each of the following pairs of expressions:

- a)  $\text{SIN}(30 + 90)$  and  $\text{COS}(30)$
- b)  $\text{TAN}(30 + 90)$  and  $-1/(\text{TAN}(30))$
- c)  $\text{SIN}(30)$  and  $\text{SQR}(1-(\text{COS}(30))^2)$

2) The Law of Sines is useful when dealing with triangles which do not necessarily contain a right angle. For such triangles

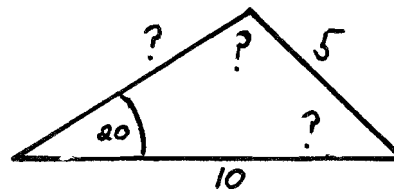
$$\frac{\text{SIN}(A)}{a} = \frac{\text{SIN}(B)}{b} = \frac{\text{SIN}(C)}{c}$$



where A, B and C are angles a, b and c are lengths. Write a program in which angle A and sides a and c are used as input to calculate and print angles B and C and side b.

```

RUN
ANGLE A? 20
LENGTH A? 5
LENGTH C? 10
ANGLE B= 116.840069
ANGLE C= 43.159931
LENGTH B= 13.0448849
    
```



3) For an angle in radians, A, the computer can evaluate  $\text{SIN}(A)$ . If the result is N, then

$$\text{SIN}(A) = N$$

For the case where the angle is unknown we must find A from

$$A = \text{ASN}(N)$$

where ASN (arcsine) is the inverse operation of SIN. The arcsine function is not available on the computer, but an alternative means is presented in the text.

Use the fact that

$$\text{TAN}(A) = \frac{\text{SIN}(A)}{\text{COS}(A)}$$

and the relation

$$\text{SIN}(A)^2 + \text{COS}(A)^2 = 1$$

to derive the expression

$$\text{ASN}(A) = \text{ATN}(A/\text{SQR}(1-A^2))$$

- 4) Write a program that will calculate and compare each of the following pairs of expressions:
- a)  $\text{LN}(3/2)$  and  $\text{LN}(3) - \text{LN}(2)$
- b)  $\text{LN}(2^{\uparrow 1.5})$  and  $1.5 * \text{LN}(2)$
- 5) Use a looping trial and error method to find the number e which makes the following equation true:

$$\text{LN}(e) = 1$$

Have your answer accurate to within .01.

- 6) Evaluate each of the following expressions for  $X=1.5, 2, 5.5, 3$ :
- a)  $\text{LN}(2.718281^{\uparrow X})$
- b)  $\text{EXP}(X) - 2.71828^{\uparrow X}$
- 7) For a population which doubles at a uniform rate (a geometrical progression), it can be shown that the doubling time, D, is

$$D = \frac{t \text{ LN}(2)}{\text{LN}(P_f) - \text{LN}(P_i)}$$

where  $P_i$  is the initial population size and  $P_f$  is the final population size after t time has elapsed.

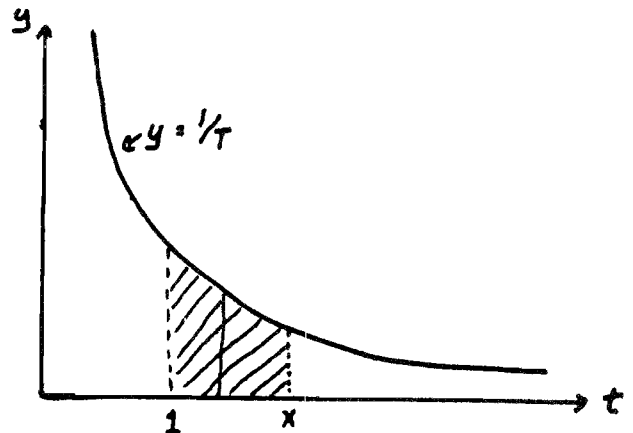
Suppose that there were 500 organisms at the beginning of a 60 minute time interval. If there are 4000 organisms at the end of this period of the time, find the doubling time.

```
RUN
THE ORGANISM'S DOUBLING
TIME WAS 20 MINUTES
```

- 8) Study the graph at right. The 'area under the curve' between 1 and X is given by  $\text{LN}(X)$ .

The area under the curve is approximately the shape of a trapezoid. The area of a trapezoid can be found from the trapezoid rule.

$$A = .5(a + b)h$$



Use the trapezoid rule to find the area under the curve when  $X=2$ . You will have to calculate  $b$  using the equation of the curve ( $Y=1/T$ ). Compare your result with  $\text{LN}(2)$ .

```

RUN
ENTER T1,T2? 1,2
TRAPEZOID      LN(2)

.75            .693147181

```

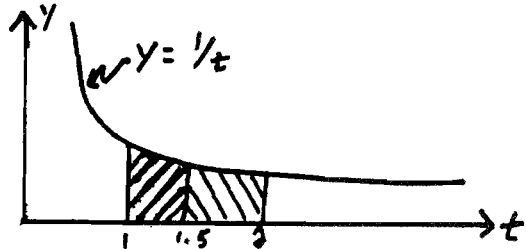
- 9) To improve the approximation given in the previous problem, the area under the curve may be subdivided into many trapezoids. Try using two trapezoids instead of one:

```

RUN
TRAPEZOID      LN(2)

.708333333    .693147181

```



- Calculate the value of  $Y$  when  $T=1.5$ .
- Calculate the area of the two trapezoids illustrated at right.
- Add the areas.
- Is the result a better approximation than that of problem 8?
- Write a program that can break the area under the curve into 10 trapezoids? 1000 trapezoids?

```

RUN
ENTER START POINT,END POINT? 1,2
ENTER # OF TRAPEZOIDS? 100
TRAPEZOID      LN(2)
.693153434    .693147181

```

# Chapter 7 Supplementary Problem Solutions

- 1) 

```
10 PI = 3.14159265
20 PRINT "TEST 1",, "TEST 2":PRINT
30 PRINT SIN((30 + 90) * PI / 180),COS(30 * PI / 180)
40 PRINT TAN((30 + 90) * PI / 180),-1 / TAN(30 * PI / 180)
50 PRINT SIN(30 * PI / 180),,SQR(1 - (COS(30 * PI / 180))↑2)
```
  
- 2) 

```
10 DEF FN ASN(X) = ATN(X / SQR(1 - X↑2))
20 INPUT "ANGLE A";AA
30 INPUT "LENGTH A";LA
40 INPUT "LENGTH C";LC
50 AC = FN ASN(LC * SIN(AA * 3.1415 / 180) / LA) * 180 / 3.1415
60 AB = 180 - AA - AC
70 LB = SIN(AB * 3.1415 / 180) * LA / SIN(AA * 3.1415 / 180)
80 PRINT "ANGLE B=";AB
90 PRINT "ANGLE C=";AC
100 PRINT "LENGTH B=";LB
```
  
- 3) 

```
SIN↑2(X) + COS↑2(X) = 1
COS(X) = SQR(1 - SIN↑2(X))
TAN(X) = SIN(X) / COS(X)
TAN(X) = SIN(X) / SQR(1 - SIN↑2(X))
X = ARCTAN(SIN(X) / SQR(1 - SIN↑2(X)))

THEREFORE, IF SIN(X) = A,
ASN(A) = ARCTAN(A / SQR(1 - A↑2))
```
  
- 4) 

```
10 PRINT "TEST #1",, "TEST #2"
20 PRINT LOG(3/2), LOG(3)-LOG(2)
30 PRINT LOG(2↑1.5),1.5 * LOG (2)
```
  
- 5) 

```
10 E = 2
20 E1 = INT(LOG(E)*100+.5)/100
30 IF E1 = 1.00 THEN PRINT "THE APPROXIMATION OF E =";
   INT(E*100+.5)/100:END
40 E = E + .001
50 GOTO 20
```



```

6) 10 PRINT"LN(2.71828↑X)", "EXP(X)-2.71828↑X"
    20 READ X
    30 IF X = 999 THEN END
    40 PRINT LOG(2.71828↑X),EXP(X)-2.71828↑X
    50 GOTO 20
    60 DATA 1.5,2,2.5,3
    70 DATA 999

7) 10 PRINT "ENTER TIME,START POP., ENDING POP."
    20 INPUT T,P1,P2
    30 D = (T * LOG(2))/(LOG(P2) - LOG(P1))
    40 PRINT
    50 PRINT "THE ORGANISIM'S DOUBLING"
    60 PRINT "TIME WAS";D;"MINUTES"

8) 10 DEF FN Y(T) = 1 / T
    20 INPUT "ENTER T1,T2";T1,T2
    30 A = FN Y(T1); B = FN Y(T2); H = T2 -
    T1
    40 X = .5 * (A + B) * H
    50 PRINT "TRAPEZOID", "LN(2)";PRINT
    60 PRINT X,LOG(T2)

9a) 10 DEF FNY(T)=1/T
    20 INPUT "ENTER T1,T2";T1,T2
    30 T3=(T1+T2)/2
    40 A1=FN Y(T3)
    50 A=FN Y(T1);B=FN Y(T2)
    60 X1=.5*(A+A1)*(T3-T1);X2=.5*(A1+B)*(T2-T3)
    70 X=X1+X2
    80 PRINT "TRAPEZOID", "LN(2)";PRINT
    90 PRINT X,LOG(T2)

9e) 10 DEF FNY(T)=1/T
    20 INPUT "ENTER START POINT,END POINT";T1,T2
    30 INPUT "ENTER # OF TRAPEZOIDS";N
    40 H=(T2-T1)/N
    50 FOR I=T1 TO (T2-H) STEP H
    60 A=.5*(FNY(I)+FNY(I+H))*H
    70 Z=Z+A
    80 NEXT I
    90 PRINT "TRAPEZOID", "LN(2)";
    100 PRINT Z,LOG(2)

```



# Chapter 7 Test

Part I Determine the exact output of each of the following programs:

- a)    10 X = -3  
      20 PRINT ABS (X)  
      30 PRINT SGN (X)
- b)    10 FOR I = 1 TO 3  
      20    READ X  
      30    PRINT SQR (X)  
      40 NEXT I  
      50 DATA 1,4,121
- c)    10 DEF FN A(N) = N \* 2 ↑ N  
      20 FOR I = 1 TO 4  
      30    PRINT FN A(I + 1) - FN A(1)  
      40 NEXT I
- d)    10 PRINT SIN (0)  
      20 PRINT COS (0)  
      30 PRINT SIN (3.141592655)
- e)    10 PRINT EXP (0)  
      20 PRINT LOG (1)  
      30 PRINT LOG (2.71828183)

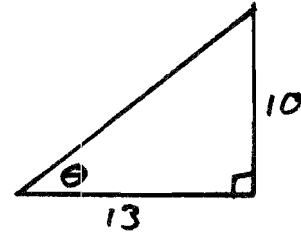
Part II Circle the programming error(s) within each program given. Explain what must be done to correct the error. If no error exists, state so.

- a)    10 DEF FN X(A) = A \* 3 / (A ↑ 2)  
      20 FOR X = 1 TO 3  
      30    PRINT FN X(A)  
      40 NEXT X
- b)    10 REM FIND THE SINE OF 45 DEGREES  
      20 PI = 3.14159265  
      30 PRINT SINE(45 \* 180 / PI)
- c)    10 REM CREATE TABLE FOR LN FUNCTION  
      20 PRINT "X", "LN(X)"  
      30 FOR X = 0 TO 10  
      40    PRINT X, LN(X)  
      50 NEXT X
- d)    10 FOR I = 1 TO 4  
      20    READ X  
      30    PRINT SQR(X - 1)  
      40 NEXT I  
      50 DATA 0,1,5,10

Part III Write each of the following programs as concisely as possible:

- a) Write a program which will find for a right triangle:
- i. The unknown angle by using the ATN function.
  - ii. The hypotenuse by using the Pythagorean theorem.

```
]RUN  
ENTER KNOWN SIDES, A & B? 10,13  
THE MISSING ANGLE = 37.568592  
THE HYPOTENUSE = 16.4012195
```



- b) Consider the following function:

$$Y = \frac{(-1)^{X-1}}{X}$$

Create a user-defined function and apply it to find the sum of the Y-values generated as X takes on the values 1,2,3,...,100.

# Chapter 7 Test Solutions

The credit for each problem is given in brackets [ ].

## Part I [ 4@ = 20 ]

- |        |        |
|--------|--------|
| a) RUN | d) RUN |
| 3      | 0      |
| -1     | 1      |
|        | 0      |
| b) RUN | e) RUN |
| 1      | 1      |
| 2      | 0      |
| 11     | 1      |
| c) RUN |        |
| 6      |        |
| 22     |        |
| 62     |        |
| 158    |        |

## Part II [ 5@ = 20 ]

- a) Line 30 should read  
30 PRINT FN X(X)
- b) Line 30 should read  
30 PRINT SIN(45 \* PI / 180)
- c) Line 30: LN(0) is non-existent.  
Line 40 all the LNs' should read LOG.
- d) Line 20 should be  
20 PRINT SQR(X - 1)  
also, SQR(0 - 1) is undefined.

## Part III

- a) 10 PI = 3.141592654 [ 30 ]  
20 INPUT "ENTER KNOWN SIDES, A & B";A,B  
30 T1 = ATN(A / B)  
40 T = T1 \* 180 / PI  
50 C = SQR(A ↑ 2 + B ↑ 2)  
60 PRINT "THE MISSING ANGLE =" ; T  
70 PRINT "THE HYPOTENUSE =" ; C
- b) 10 DEF FN Y(X) = ( - 1 ↑ (X - 1)) / X [ 30 ]  
20 FOR X = 1 TO 100  
30 P = FN Y(X)  
40 T = T + P  
50 NEXT X  
60 PRINT "THE TOTAL =" ; T





	16384	8192	4096	2048	1024	512	256	128	64	32	16	8	4	2	1
0	0	0	0	0	1	0	1	0	1	0	0	0	1	1	0

Notice that the decimal value of the MSB taken alone is 5. The value of the LSB taken alone is 70. The integer represented can also be found from  $5 * 256 + 70$ .

The left-most bit determines the algebraic sign of the integer. If the left-most bit is a 0, the integer is positive, and if it is 1, then the integer is negative. The largest positive integer available is formed when the other 15 are all set to 1: +32767. Representation of negative integers is more complicated than positive, so this topic will be covered at the end of this lesson.

A real number is stored in five adjacent bytes; four of them contain the mantissa, while the fifth holds the exponent. The real number is constructed from the mantissa and exponent according to

$$\text{mantissa} \times 2^{\text{exponent}}$$

The bits of the mantissa do not correspond directly to powers of two. Converting a real number from binary to decimal is beyond the scope of this text.



# Worksheet 8.1

Part I Determine the ASCII character which could be represented by each of the following bytes using the table on Page 8.3 of the text:

	<u>128</u>	<u>64</u>	<u>32</u>	<u>16</u>	<u>8</u>	<u>4</u>	<u>2</u>	<u>1</u>
a.	0	1	0	0	0	0	0	0
b.	0	0	1	0	0	0	0	1
c.	0	0	1	1	1	1	1	1
d.	0	0	1	1	0	0	0	1
e.	0	1	0	1	1	0	1	0

Part II Determine the binary bit pattern of the ASCII code for each of the following:

- a space
- the letter 'C'
- the number '9'

Part III Determine the output of the following programs:

- ```
10 FOR I = 1 TO 3
20   PRINT CHR$(I + 82);
30 NEXT I
40 PRINT CHR$(68)
50 END
```
- ```
10 N$ = "C": M$ = "COMMODORE"
20 PRINT ASC (N$)
30 PRINT ASC ("N$")
40 PRINT ASC (M$)
50 END
```

# Worksheet 8.1 Solutions

- Part I
- a. @
  - b. I
  - c. ?
  - d. l
  - e. z

- Part II
- a. 0 0 1 0 0 0 0 0 = 32
  - b. 0 1 0 0 0 0 1 1 = 67
  - c. 0 0 1 1 1 0 0 1 = 57

- Part III
- a. STUD
  - b. 67  
78  
67

Note: ASC only finds the ASCII code of one character, the first character in the string. Hence, ASC ("C") is equal to ASC ("COMMODORE"). MID\$ must be used to isolate letters within a string for the ASC function.

# Lesson 8.2

OBJECTIVES: a) to apply the string manipulation function LEFT\$, RIGHT\$, MID\$ and LEN

b) to convert numeric values to string values using STR and to convert string values into numeric values using VAL

---

ASSIGNMENT: Read pages 8.6 through 8.8  
Text problems 1,5,7,9,13,21,23  
Homework problems 6,8,12,14,16,20,24

---

The string manipulation functions are important for writing programs which handle words. Using these functions, it would be possible to write a crude word processing program. Actual word processing programs are written in assembly language so that they will run faster and more efficiently.

Note that '+' is a valid function for strings. For example:

```
10 A$ = "OUT"  
20 B$ = "PUT"  
30 C$ = A$ + B$  
40 PRINT C$  
50 END
```

```
RUN  
OUTPUT
```

This function is used in Worksheet 8.2 Problem 4.

First, review each of the functions in the table on page 8.6 of the text, and then slowly review problems 8.3 and 8.4. In reviewing the table, be careful to note that the N2 in M\$ = MID\$ (A\$, N1, N2) specifies the length of the substring, not the position of the last character.

To assist you in reviewing Programs 8.3 and 8.4 an analysis follows below:

## PROGRAM 8.3

```
LINE 40: LEN(T$) returns an integer equal to the number of  
characters found in T$. The FOR...NEXT loop is executed  
LEN(T$) times since it goes from LEN(T$) to 1 STEP -1.  
LINE 50: Prints a string with the length of 1, starting at  
position X in T$.  
LINE 60: Completes the FOR...NEXT loop initiated in line 40.  
LINE 70: Acts as a line feed to negate the semi-colon from the  
last print in the FOR...NEXT loop.  
LINE 80: VAL(T$) converts the first set of numeric characters in  
T$ into a numeric value, which is then printed.  
LINE 90: Prints the value of T$ and a message.
```

LINE 100: The first argument in the print statement takes the rightmost half (LEN(T\$)/2) characters of T\$ and prints them. The second argument prints the left half.  
 LINE 110: Prints the value of the first set of numeric characters found and a message.  
 LINE 120: Initiates a FOR...NEXT loop. The starting value is found by taking the VAL(T\$) <see line 80> and reconverting that value to a string. The length (LEN(X\$)) of that string is used as the counter variable.  
 LINE 130: Prints a string 1 character in length from the string derived from the value of T\$ starting at position X.  
 LINE 140: Completes the FOR...NEXT loop.

#### PROGRAM 8.4

LINE 10: Sets up a FOR...NEXT loop to read a randomly assigned number of words from a DATA statement.  
 LINE 20: Reads a string (W\$) from DATA statement (lines 610-620).  
 LINE 30: Completes loop initiated in 10.  
 LINE 40: Tells the user that the computer is ready.  
 LINE 50: C acts as a flag. If a correct guess is made in line 110, then C will be set to zero. If an incorrect guess is made, C will retain its starting value of one.  
 LINE 60: Initiates a loop with the loop variable going as long as W\$ and loads it into D\$.  
 LINE 70: Sets up a string of question marks as long as W\$ and loads it into D\$.  
 LINE 80: Completes the loop started in 60  
 LINE 90: Prints D\$ to the output screen.  
 LINE 100: Prints message to user telling the number of incorrect guesses.  
 LINE 110: Prompts and accepts a guess (L\$).  
 LINE 210: Initiates a loop from 1 to the length of W\$  
 LINE 220: Tests to see if the guess letter (L\$) is not equal to any of the characters in W\$. If so, we go to line 270.  
 LINE 230: If the text finds out our string (L\$) is in W\$, we set our mistake flag (C) to 0.  
 LINE 240: If this is the first time through the loop, we set L\$ to the first character of D\$.  
 LINE 250: If this is the last time through the loop, we set L\$ equal to the last character of D\$.  
 LINE 260: If neither of the above conditions is correct, we set L\$ equal to the Ith character of D\$.  
 LINE 270: Completes the loop started in 210.  
 LINE 280: Checks to see if D\$ is equal to W\$; if so, goes to line 400.  
 LINE 290: Updates the mistake counter and reinitializes the mistake flag.  
 LINE 300: Checks to see if we have too many mistakes; if so, goes to line 500.  
 LINE 310: Returns to line 90  
 LINE 400: If W\$ equaled D\$, we win and the computer prints out the 'winners' message and stops the program.  
 LINE 500: Prints out the 'losers' message.  
 LINE 510: Tells the user what the word was.  
 LINE 610: Data line used for READ in line 20.  
 LINE 620: Data line used for READ in line 20.

# Worksheet 8.2

Part I Determine the exact output of each of the following programs:

- a) 

```
10 A$ = "PROGRAMMING"
20 PRINT LEN (A$)
30 PRINT LEFT$ (A$,3)
40 PRINT MID$ (A$,3,2)
50 PRINT RIGHT$ (A$,2)
```
- b) 

```
10 A$ = "RESPECT"
20 B$ = "COINCIDE"
30 C$ = "STIFLE"
40 PRINT MID$ (A$,4,2) + LEFT$ (C$,3) + RIGHT$ (B$,4)
```
- c) 

```
10 Q$ = "NITROGEN"
20 FOR I = 3 TO 7 STEP 2
30   PRINT MID$ (Q$,I,1);
40 NEXT I
```
- d) 

```
10 N$ = "123"
20 T$ = MID$ (N$, LEN (N$),1)
30 V = VAL (T$) + 1
40 S$ = STR$ (V)
50 M$ = N$ + S$
60 PRINT M$
70 IF V <= 6 THEN N$ = M$: GOTO 20
```
- e) 

```
10 W$ = "HOTDOG"
20 FOR X = 1 TO 5
30   IF LEFT$ (W$,X) > RIGHT$ (W$,X) THEN PRINT "LEFT ";
40   IF LEFT$ (W$,X) < RIGHT$ (W$,X) THEN PRINT "RIGHT ";
50 NEXT X
```

Part II Write a program that will analyze a user's input and tell him or her whether the input was numeric or alphanumeric.

```
RUN
?14254
THE INPUT WAS NUMERIC
```

```
RUN
?1K42PO54
THE INPUT WAS ALPHANUMERIC
```

# Worksheet 8.2 Solutions

## Part I

- a) 11  
PRO  
OG  
NG
- b) PESTICIDE
- c) TOE
- d) 1234  
12345  
123456  
1234567
- e) LEFT RIGHT LEFT RIGHT RIGHT

Part II

```
10 INPUT A$
20 IF STR ( VAL (A$)) = A$ THEN PRINT "THE INPUT WAS
    NUMERIC": END
30 PRINT "THE INPUT WAS ALPHANUMERIC"
```

# Chapter 8 Supplementary Problems

- 1) Write a program which will accept a string entered by the user and then print out the number of occurrences of each letter in the string.
- 2) Write a program which will accept a sentence with spaces moved from the beginning of each word to some random position within each word. One letter words should be excluded.
- 3) The following program searches for the substring S\$ within the phrase C\$. Unfortunately, the following program can only detect the first occurrence of the substring. As shown below, the substring "OU" is found once when, in fact, it occurs twice. Modify the program so that it locates all occurrences of a substring:

```
10 C$ = "I THINK YOUR FOOLISH FATHER IS OUTSIDE TAKING
    A BATH"
20 INPUT "SUBSTRING TO SEARCH FOR: ";S$
30 E = LEN (C$) - LEN (S$) + 1
40 FOR I = 1 TO E
50     IF MID (C$,I, LEN (S$)) = S$ THEN PRINT "SUBSTRING
        FOUND STARTING AT POSITION ";I: GOTO 80
60 NEXT I
70 PRINT "STRING NOT FOUND"
80 END
```

```
RUN
SUBSTRING TO SEARCH FOR: OU
STRING FOUND STARTING AT POSITION 10
```

- 4) Write a program which allows the user to play a word guessing game. Follow these specifications:
  - a) Put ten 'secret' words of various lengths into a data statement.
  - b) Select one word at random from the data statement.
  - c) Tell the player the length of the word.
  - d) Reveal a letter from a random position in the word and tell the player which position has been revealed.
  - e) Allow the player to take a guess.
  - f) If the player does not guess the word, add one point to his score and reveal another letter. Be careful not to reveal a letter which has already been used as a clue.
  - g) Repeat steps d, e and f until the player guesses the word. When the word is guessed, print the player's score and ask if he would like to play again.

# Chapter 8 Supplementary Problem Solutions

```
1) 10 DIM S$(50)
20 INPUT "ENTER A STRING";A$
30 M=1
40 FOR I=1 TO LEN(A$)
50   T$=MID$(A$,I,1)
60   FOR J=1 TO I
70     IF T$<>S$(J) THEN C=C+1
80   NEXT J
90   IF C=I THEN S$(M) = T$: M=M+1
100  C=0
110 NEXT I
120 FOR I=1 TO M
130   FOR J=1 TO LEN(A$)
140     IF S$(I) = MID$(A$,J,1) THEN K=K+1
150   NEXT J
160   PRINT S$(I); " OCCURS";K; "TIMES IN ";A$
170   K=0
180 NEXT I
190 END

2) 100 INPUT "STRING";ST$
110 LN=LEN(ST$)
120 GOSUB 1000: S1=R
130 IF S1=0 THEN 300
140 REM
200 REM * MAIN ROUTINES *
210 GOSUB 1000: S2=R-1
220 IF S2-S1 <= 1 THEN 250
230 X=INT(RND(0)*(S2-S1-1)+1)
240 ST$=LEFT$(ST$,S1-1)+MID$(ST$,S1+1,X)+MID$(ST$,S1,1)+RIGHT$(ST$,LN-S1-X)
250 S1=S2+1
260 IF S2<>LN THEN 200
270 REM
300 PRINT: PRINT ST$
999 END
1000 REM SEARCH FOR SPACE
1010 REM
1020 FOR I=LN TO (S1+1) STEP -1
1030   IF MID$(ST$,I,1) = " " THEN R=I
1040 NEXT I
1050 IF R=S1 THEN R=LN+1
1060 RETURN
```



```

3) 5 REM ENTER "ZZZ" TO STOP
10 C$="I THINK YOUR FOOLISH FATHER IS OUTSIDE TAKING A BATH"
15 C=0
20 PRINT"ENTER 'ZZZ' TO STOP"
22 INPUT "SUBSTRING TO SEARCH FOR";S$
25 IF S$="ZZZ" THEN 80
30 E=LEN(C$) - LEN(S$) + 1
40 FOR I=1 TO E
50   IF MID$(C$,I,LEN(S$))=S$ THEN PRINT "SUBSTRING FOUND STARTING AT";I:C=C+1
60 NEXT I
65 IF C>0 THEN 15
70 PRINT "STRING NOT FOUND":GOTO 15
80 END

```

```

4) 10 DIM W(10)
20 RESTORE
25 FOR I=1 TO 10:W(I)=0:NEXT I
30 S=1
35 X=INT(RND(0)*10+1)
40 FOR I=1 TO X
50   READ W$
60 NEXT I
70 PRINT "THE LENGTH OF YOUR WORD IS";LEN(W$)
80 IF S=LEN(W$) THEN 190
90 P=INT(RND(0)*LEN(W$)+1)
100 IF W(P)=1 THEN 90
110 W(P)=1
120 PRINT "THE LETTER AT POSITION";P;"IS ";MID$(W$,P,1)
130 INPUT "YOUR GUESS";A$
140 IF A$<>W$ THEN S=S+1:GOTO 80
150 PRINT "*** CORRECT ***":PRINT "YOUR SCORE IS ";S
160 INPUT "PLAY AGAIN (Y/N)";A$
170 IF A$ = "Y" THEN 20
180 END
190 PRINT "*** YOU LOSE ***":PRINT "YOUR SCORE IS";S
200 DATA DOG,CAT,COMPUTER,JULIO,IGLESIAS,SHOE,TENNIS,BEACH,RADIO,PENCIL

```



# Chapter 8 Test

Part I Determine the exact output of each of the following programs:

- a) 10 A\$ = "EXACT OUTPUT"  
20 PRINT LEN (A\$)  
30 PRINT LEFT\$ (A\$,8)  
40 PRINT RIGHT\$ (A\$,8)
- b) 10 A\$ = "BYTE SIZE"  
20 FOR I = 2 TO 6 STEP 2  
30 PRINT MID\$ (A\$,I,1)  
40 NEXT I
- c) 10 FOR J = 70 TO 88 STEP 9  
20 PRINT CHR\$(J);  
30 NEXT J
- d) 10 A\$ = "AX"  
20 PRINT ASC ( LEFT (A\$,1)) +  
ASC (RIGHT\$ (A\$,1))
- e) 10 A\$="20 DOLLARS AND 19 CENTS"  
20 D = VAL ( LEFT\$ (A\$,12))  
30 C = VAL ( RIGHT\$ (A\$,12))  
40 PRINT (D \* 100 + C) / 100

Part II Circle the programming error(s) within each program given. Explain what must be done to correct the error. If no error exists, state so.

- a) 10 REM PRINT A\$ BACKWARDS  
20 INPUT A\$  
30 FOR I = LEN \$(A\$) TO 1  
40 PRINT MID(A\$,1,I);  
50 NEXT I
- b) 10 REM REMOVE 19 FROM A  
FOUR-DIGIT YEAR  
20 X=1983  
30 S\$ = SQR(X)  
40 PRINT LEFT(S\$,2)
- c) 10 REM DETERMINE # OF VOWELS IN A\$  
20 INPUT A\$  
30 FOR I = 1 TO LEN \$(A\$)  
40 FOR J = 1 TO 5  
50 READ V\$  
60 IF MID\$ (A\$,I) = V\$ THEN N = N + 1  
70 NEXT J  
80 NEXT I  
90 PRINT "NUMBER OF VOWELS =";N  
100 DATA A,E,I,O,U
- d) 10 REM CONVERT ASCII  
CHARACTER INTO BINARY  
20 INPUT A\$:D = ASC (A\$)  
30 FOR I = 8 TO 1 STEP -1  
40 K = 2 ↑ I  
50 IF D / K >= 1 THEN  
B(I) = 1:D = D - K  
60 NEXT I  
70 REM  
80 FOR I = 1 TO 8  
90 PRINT B(I)  
100 NEXT I

Part III Write each of the following programs as concisely as possible:

- a) Each letter of a message has been altered by changing the letter to ASCII code, subtracting a number given by its position in the message, then reconvertting the result to an ASCII character. For example, "HI" would be encoded as "GG". 1 was subtracted from the ASCII code for H because H is the first letter in the message. 2 was subtracted from the ASCII code for I because I is the second letter in the message.

Write a program to decode the following message:  
LCBPDHI9I?H

- b) Write a program which will produce an output similar to the following for a sentence entered at the keyboard. Don't allow sentences to be entered having more than 39 characters. You may use only one PRINT statement.

```
RUN
?I SEEM TO KEEP FORGETTING ONE WORD
I SEEM TO KEEP FORGETTING ONE
I SEEM TO KEEP FORGETTING
I SEEM TO KEEP
I SEEM TO
I SEEM
I
```

# Chapter 8 Test Solutions

The credit for each problem is given in brackets [ ].

Part I [ 5@ = 25 ]

- |    |                                   |    |            |
|----|-----------------------------------|----|------------|
| a) | RUN<br>12<br>EXACT OU<br>T OUTPUT | c) | RUN<br>FOX |
| b) | RUN<br>Y<br>E<br>S                | d) | RUN<br>153 |
|    |                                   | e) | RUN<br>20  |

Part II [ 5@ = 20 ]

- a) Line 30 should be 30 FOR I = LEN (A\$) TO 1 STEP -1  
Line 40 should be 40 PRINT MID\$ (A\$,I,1)
- b) Line 30 should be 30 S\$ = STR\$(X)  
Line 40 should be 40 PRINT RIGHT\$ (S\$,2)
- c) Line 30 should be 30 FOR I = 1 TO LEN (A\$)  
Line 40 should be 60 IF MID\$ (A\$,I,1) = V\$ THEN N = N + 1
- d) Line 40 should be 40 K = 2 ↑ (I - 1)

Part III

- a) 10 DIM AC(39) [ 30 ]  
20 INPUT ST\$  
30 PRINT  
40 FOR I = 1 TO LEN (ST\$)  
50 AC(I) = ASC ( MID\$ (ST\$,I,1))  
60 AC(I) = AC(I) + I  
70 PRINT CHR\$ (AC(I))  
80 NEXT I  
90 END
- b) 10 INPUT ST\$ [ 25 ]  
20 LN = LEN (ST\$)  
30 IF LN > 39 THEN 10  
40 FOR I = LN TO 1 STEP - 1  
50 IF MID\$ (ST\$,I,1) = " " THEN PRINT LEFT\$ (ST\$,I-1)  
60 NEXT I  
70 END



# Lesson 9.1

- OBJECTIVES:
- a) to understand the concept of files and how to implement them using the cassette recorder
  - b) to learn how to manipulate files using the cassette recorder, the commands OPEN and CLOSE, and the statements PRINT# and INPUT#
  - c) to understand how a sequential file, such as a cassette file, is set up and is used
- 

ASSIGNMENT: Read pages 9.1 through 9.5 Do review problems 1,2  
Text problems 2,4 Homework problems 1,3,5

---

When a programmer begins to write programs that use a large amount of data and that use the data in separate programs, it becomes obvious that retyping lengthy READ...DATA lines is a poor solution. Understanding and implementing files will solve this problem. In essence, a file is a set of data that has been given a name so that it can be accessed by several programs. There are two major places to store files: cassette tapes and diskettes. The rest of this lesson explains cassette files; diskette files will be covered in a later lesson.

---

In order to create a cassette file, you must have a cassette tape. A leaderless tape is recommended, but any tape will suffice. The first step in creating a file on cassette tape is to position the tape to an empty space on the tape. If your recorder has a tape counter, save the number marking the position of the file. Later, when you want to access the data in the file, the tape should be positioned just before this number. Although the computer will search the entire side of a cassette for a file, this generally takes a long time. By positioning the tape correctly, the time that the computer uses to search can be drastically reduced.

After setting the tape to a blank part of tape, you are ready to start using files. Before any work with files can be done, it is necessary to OPEN the file. This command uses the following format:

```
OPEN <file#>, <device#>, <mode>, "<file name>"
```

The meanings of the various device numbers and file numbers are explained well in the text. The mode number is, however, worthy of further comment. If the mode number specified is 0, the computer will ask the user to hit PLAY on the tape recorder. The screen will clear while the program is looking through the cassette. The computer will try to find a file on the remaining portion of the tape with the name contained in the OPEN command. If the computer cannot find the specified file after reading through the entire tape, it will stop. You can get the screen back by hitting the RUN STOP key.

After a file is OPENed, it must be CLOSED to insure that the data in it is stored correctly. The CLOSE statement also cuts the channel of communication between the file and the current program, which saves memory space. Remember to use the file number when closing a file, not the device number or the mode number.

At this point, students may be itching to send information to and retrieve information from a file. This is accomplished using the statements PRINT# and INPUT#.

The PRINT# statement is used to send information to a file. It acts in the same manner as a PRINT statement with the sole exception that the computer is instructed to print to a file as opposed to printing on the monitor. The standard format for printing to a file looks like this:

```
PRINT#1, A$;" ";B$;" ";C
```

The commas are used to separate the fields within the record and to allow easy retrieval by the INPUT# statement.

The INPUT# statement is used to retrieve information from a file. The INPUT# used to retrieve the information put into the file by the above PRINT# command would look like this:

```
INPUT#1,A$,B$,C
```

---



The text gives a simple explanation of how to use cassette files. The material that follows gives a considerably more detailed explanation and should be covered only if your students will be making considerable use of files. You might consider photocopying this lesson as a hand-out for student reference.

A sequential file stores information as a text file, which means that the data is stored as the ASCII codes for each individual character. The ASCII codes are stored in records with each record terminated by a RETURN character (↵), (ASCII code 13). In any record, there can be several fields, which are separated by commas (ASCII code 44). For example, the data stored in the file CREATE by Program 9.1 on page 9.3 is stored as follows:

G	W	E	N		W	A	L	D	E	N	,	4	.	4	5	,	3	9	↵				
S	A	N	D	Y		B	E	C	K	E	R	M	A	N	,	4	.	7	5	,	4	2	↵
R	U	B	E	N		S	T	E	C	K	,	5	.	1	0	,	3	6	.	7	5	↵	
R	U	T	H		O	B	R	E	,	4	.	9	5	,	2	7	.	5					eof

Each of the lines shown above containing a name, wage and hours worked is a single record which is terminated by a RETURN character (↵). The commas separate the fields within a single record and are used by the Commodore so that it can recognize each item as it is read back out of the file. At the very end of the file the computer places an end-of-file (eof) marker so that when data is read, the computer will be able to determine where the file ends.

Rather than placing each of the three items of data (name, hours worked, wage) in a single record it is possible to place each item in a separate record. To do this, replace line 60 of program 9.1 with the following:

```
60 PRINT#1,N$
62 PRINT#1,W
64 PRINT#1,H
```

Now each record contains only a single field.

G	W	E	N		W	A	L	D	E	N	↵				
4	.	4	5	↵											
3	9	↵													
S	A	N	D	Y		B	E	C	K	E	R	M	A	N	↵

4	.	7	5	←
---	---	---	---	---

4	2	←
---	---	---

R	U	B	E	N		S	T	E	C	K	←
---	---	---	---	---	--	---	---	---	---	---	---

5	.	1	0	←
---	---	---	---	---

3	6	.	7	5	←
---	---	---	---	---	---

R	U	T	H		O	B	R	E	←
---	---	---	---	--	---	---	---	---	---

4	.	9	5	←
---	---	---	---	---

2	7	.	5	eof
---	---	---	---	-----

To read this new file, make the following changes in Program 9.2:

```
40 INPUT#2,N$
42 INPUT#2,W
44 INPUT#2,H
```

Notice that again the variable W must be input from the file, as it is in Program 9.2, even though it will not be printed at line 50. This is because the computer reads across the file item by item and without line 42, the wage (W) would be substituted for the hours worked (H).

Of the two approaches to storing the name, wage and hours worked in a sequential file, neither is necessarily preferable, but once you have decided how the data is to be stored, the program reading the data back out of the file must be written to reflect the proper file structure.

# Worksheet 9.1

Part I Circle the programming error(s) within each program given. Explain what must be done to correct the error. If no error exists, state so.

- a) This program should create a cassette file named PHONE and allow the user to enter a list of names and telephone numbers:

```
10 OPEN 1,1,0,"PHONE"
20 FOR X=1 TO 5
30 INPUT "NAME: ";N$
40 INPUT "TELEPHONE NUMBER: ";P$
50 PRINT#0,N$;" ";P$
60 NEXT X
70 CLOSE "PHONE"
```

- b) This program should print a table listing the names and telephone numbers contained in the file PHONE:

```
10 OPEN 65,1,0,"PHONE"
20 FOR C=1 TO 5
30 PRINT "NAME";"NUMBER"
40 PRINT "----";"-----"
50 INPUT#1,A$,B$
60 PRINT A$,B$
70 NEXT C
80 CLOSE 65
```

Part II After the program that appears directly below is run (and the tape is rewind), determine the exact output of each of the following programs:

```
10 OPEN 4,1,1,"NUMBERS"
20 FOR N=3 TO 10 STEP 2
30 X = 2*N + X
40 PRINT#4,X
50 NEXT N
60 CLOSE 4
```

- a) 

```
10 OPEN 1,1,0,"NUMBERS"
20 FOR Q=3 TO 10 STEP 2
30 INPUT#1,E
40 L=L+E
50 NEXT Q
60 PRINT L
70 CLOSE 1
```

- b) 

```
10 OPEN 7,1,0,"NUMBERS"
20 FOR P=2 TO 5
30 PRINT M
40 INPUT#7,M
50 NEXT P
60 CLOSE 7
```

# Worksheet 9.1 Solutions

Part I a) Line 10 should read

```
10 OPEN 1,1,1,"PHONE"
```

or

```
10 OPEN 1,1,2,"PHONE"
```

Line 50 should read

```
50 PRINT#1,N$;" ";P$
```

Line 70 should read

```
70 CLOSE 1
```

b) Instead of being at line 20, the FOR..NEXT loop should start at line 45 so that the heading is not printed each time through the loop. Also, line 50 should read

```
50 INPUT#65,A$,B$
```

Part II

a) 100

b) 0  
6  
16

## Lesson 9.2

OBJECTIVES: a) to learn how to update files stored on cassette  
b) to understand the STATUS variable and its uses

---

ASSIGNMENT: Read pages 9.5 to 9.13                      No review problems  
                  No text problems                              No homework problems

---

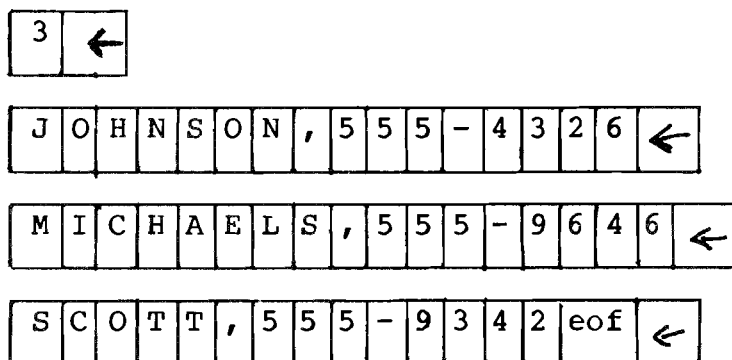
Although editing a cassette file is generally a messy, time-consuming process, it can be done, and the text covers this well. When appending records to the end of a file, it is important to know the number of records that are already there. This problem can be solved in two different ways.

When a record from a file is read, the computer always checks ahead to see whether there exists another record following it. The system variable STATUS allows us to determine whether the end of a file has been reached. The comparison (STATUS AND 64) = 64 will be true if the end of the file has been reached. If the above comparison is to be used, it must occur directly after an INPUT# which found a record.

Another method of finding the end of a file is to know the number of records that it holds. This number can be stored in the first record of your file. For example, consider the following program:

```
10 OPEN 1,1,1,"EXAMPLE"
20 INPUT "HOW MANY RECORDS IN THIS FILE:";N
30 PRINT#1,N
40 FOR X=1 TO N
50   PRINT "RECORD #";X
60   INPUT "NAME:";N$
70   INPUT "PHONE NUMBER:";T$
80   PRINT#1,N$;" ";T$
90 NEXT X
100 CLOSE 1
```

This program stores names and phone numbers in a file called EXAMPLE. The first record consists of a number representing the number of names in the file. Schematically the file might look like this:



Now, when a list of all the names and corresponding phone numbers is desired, the variable STATUS need not be used. The first record is read, and the value found there is used in a FOR..NEXT loop. The program below illustrates this:

```
10 OPEN 1,1,0,"EXAMPLE"  
20 INPUT#1,N  
30 PRINT "THERE ARE";N;"NAMES IN THE FILE."  
40 FOR X=1 TO N  
50   INPUT#1,P$,T$  
60   PRINT N$;TAB(20);T$  
70 NEXT X  
80 CLOSE 1
```

This method allows the programmer to know the number of items in a file without going through the whole file. Especially in larger programs, the process of reading through the whole file to find the length can be rather time-consuming.

Note that the number in the first record is not the number of records in the file. It is rather the number of records which follow, since the record that contains the number is not considered because it does not contain any of the data the file stores.

# Worksheet 9.2

Part I Write each of the following programs as concisely as possible:

There exists a file named "BIRDS" on a cassette tape which has an unknown number of records. Each record contains two fields. The first field is occupied by a bird's name (B\$). The second field is occupied by an integer which represents the number of sightings of that bird by local ornithologists (S). The file was created with the following short program:

```
10 OPEN 35,1,1,"BIRDS"  
20 INPUT "BIRD";B$  
30 INPUT "NUMBER OF SIGHTINGS";S  
40 PRINT#35,B$;" ";S  
50 INPUT "ANOTHER (Y/N)";Y$  
60 IF Y$ = "Y" THEN 20  
70 CLOSE 35  
80 END
```

a) Write a program to create a list of all of the birds and their number of sightings. Do not allow the program to end with an error.

b) Write a program that will update the file by allowing local ornithologists to go through the list and enter any additional sightings.

c) Write a program that will allow the ornithologist to enter new records when new birds are sighted.

# Worksheet 9.2 Solutions

```
Part I  a)  10 OPEN 1,1,1,"BIRDS"
           20 INPUT#1,B$,S
           30 PRINT B$;TAB(20);S
           40 IF (STATUS AND 64) <> 64 THEN 20
           50 CLOSE 1
           60 END

        b)  10 DIM B$(15),S(15)
           20 OPEN 1,1,0,"BIRDS"
           30 X = X + 1
           40 INPUT#1,B$(X),S(X)
           50 PRINT B$(X);TAB(20);S(X)
           60 INPUT "HOW MANY ADDITIONAL SIGHTINGS";A
           70 IF A<0 THEN 60
           80 S(X) = S(X) + A
           90 IF (STATUS AND 64) <> 64 THEN 30
          100 CLOSE 1
          110 PRINT "REWIND TAPE TO THE BEGINNING OF THE FILE"
          120 INPUT "HIT <RETURN> WHEN READY ";D$
          130 OPEN 1,1,1,"BIRDS"
          140 FOR N=1 TO X
          150   PRINT#1,B$(X);", ";S(X)
          160 NEXT N
          170 CLOSE 1
          180 END

        c)  10 DIM B$(25),S(25)
           20 OPEN 1,1,0,"BIRDS"
           30 X = X + 1
           40 INPUT#1,B$(X),S(X)
           50 IF (STATUS AND 64) <> 64 THEN 30
           60 CLOSE 1
           70 X = X + 1
           80 INPUT "BIRD";B$(X)
           90 INPUT "SIGHTINGS:";S(X)
          100 IF X = 25 THEN 130
          110 INPUT "ANOTHER BIRD (Y/N)";Y$
          120 IF ASC(Y$) = 89 THEN 70
          130 PRINT "REWIND TAPE TO THE BEGINNING OF THE FILE"
          140 INPUT "HIT <RETURN> WHEN READY";S$
          150 OPEN 1,1,1,"BIRDS"
          160 FOR I=1 TO X
          170 PRINT#1,B$(I);", ";S(I)
          180 NEXT I
          190 CLOSE 1
          200 END
```



# Lesson 9.3

## LESSON 9.3

- OBJECTIVES:
- a) to learn how to use diskette files
  - b) to learn how to update files on diskettes
  - a) to understand the concepts behind the bubble sort and how to use one

---

ASSIGNMENTS:    Read pages 9.14 to 9.18                    No review problems  
                  Text problems 6,8,10                         Homework problems 7,9,11

---

Diskette files work, for the most part, in the same way as cassette files. There are several reasons, however, why diskettes are more popular than cassettes. First, the computer can create and access diskette files much faster than cassette files. If a cassette tape file exists near the end of a side of the tape, the recorder must look through all of the files on that side of the tape unless the programmer has moved the tape manually by pressing the fast-forward button. A disk drive is a random device, which means it does not have to move sequentially as a cassette recorder does; rather it can jump directly to the desired file.

Also, multiple diskette files can be OPENed and used simultaneously, which can save time. This also makes it easier to update the contents of a diskette file since an entire file need not be read into memory, where there may not be enough space for it. Examine the following example:

```
10 OPEN 2,8,2,"CATS,S,R"  
20 OPEN 3,8,3,"PETS,S,W"  
30 INPUT#2,A$  
40 S = STATUS  
50 PRINT#3,A$  
60 IF (S AND 64) <> 64 THEN 30  
70 CLOSE 2: CLOSE 3  
80 END
```

This program will copy the contents of the file CATS into the file PETS without reading the entire file CATS into memory. Line 40 is necessary because the STATUS variable will change after the PRINT# statement in line 50. The variable S stores what was in the STATUS variable for the comparison in line 60.

The third advantage of diskettes is the convenience they afford the programmer. Instead of spending your time rewinding tapes, hitting PLAY or PLAY and RECORD, you can be doing something worthwhile; the drive takes care of the repetitive, time-consuming business of finding open space in which to put a file, remembering where it was and trying to remember what is in it two weeks later.

Be careful not to use channels 0 and 1 when OPENING diskette files. These channels are reserved for the Operating System of the computer. Also, as the text suggests, to reduce confusion, it is wise to OPEN diskette files with the same file number and channel number. Once OPENed, diskette files are manipulated exactly like cassette files. Data is transferred in the same way with the statements PRINT#<file number> and INPUT#<file number>.

---

Although the example of the bubble sort in the text covers the basic concepts inherent in a bubble sort, another example will be presented here. Consider the following program:

```
10 REM
20 PRINT "INPUT 5 WORDS TO BE SORTED"
30 PRINT "USING A BUBBLE SORT"
40 PRINT
50 FOR X=1 TO 5
60   PRINT "WORD NO. ";X;
70   INPUT A$(X)
80 NEXT X
90 REM
100 REM START BUBBLE SORT
110 REM
120 FOR T=1 TO 4
130   FOR U=5 TO T+1 STEP -1
140     IF A$(U) > A$(U-1) THEN 180
150     TEMP$ = A$(U)
160     A$(U) = A$(U-1)
170     A$(U-1) = TEMP$
180   NEXT U
190 NEXT T
200 REM
210 REM PRINT OUT SORTED LIST
220 FOR C=1 TO 5
230   PRINT A$(C)
240 NEXT C
250 END
```

```
RUN
INPUT 5 WORDS TO BE SORTED
USING A BUBBLE SORT
WORD NO. 1 ?JOHN
WORD NO. 2 ?MIKE
WORD NO. 3 ?GREG
WORD NO. 4 ?BRUCE
WORD NO. 5 ?CHRIS
BRUCE
CHRIS
GREG
JOHN
MIKE
```

This program accepts 5 names, sorts them alphabetically using a bubble sort, and then prints the sorted list. The bubble sort essentially looks through the list, comparing neighboring items. If the two items are out of alphabetical order, they are switched. By going through the list enough times, an alphabetical list results.

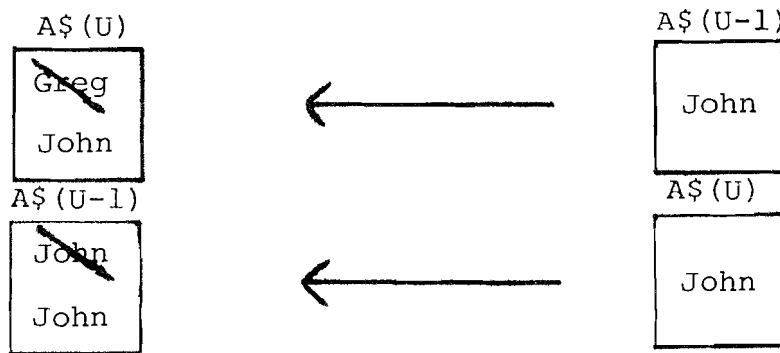
The actual switching of the two values happens between lines 150 to 170. It is important that the student understand why the variable TEMP\$ is used. This can be illustrated using boxes to depict the values contained by variables. For example, say that the two items that are out of place are JOHN and GREG:



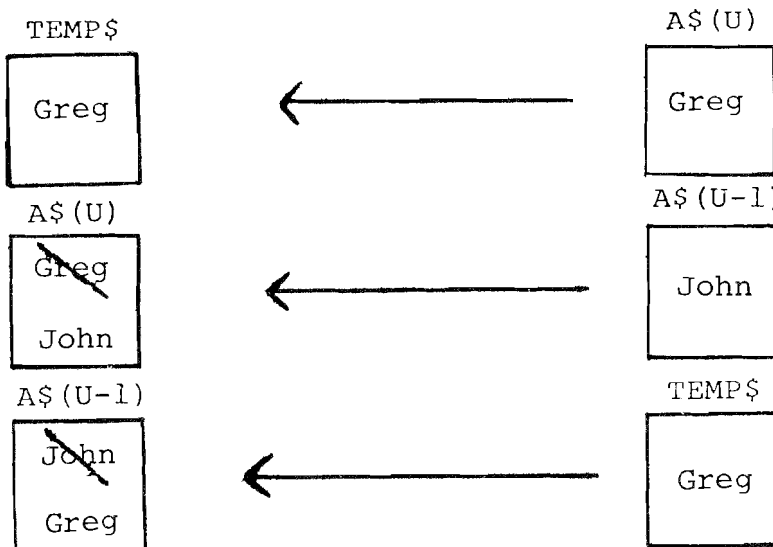
If one were to try to switch them without using a temporary "holding" string, he might use the lines

```
160 A$(U)=A$(U-1)
170 A$(U-1)=A$(U)
```

and get rid of line 150. Emphasize that a variable can hold only one value at a time and that by reassigning A\$(U-1) to A\$(U), the name that was held originally by A\$(U), "GREG", is lost. If we visualize boxes, this transaction would look something like this:



Clearly, such an outcome is not desirable. The correct attack is the one used in the program. The name GREG is held in TEMP\$ while the variable which originally contained GREG is changed. Then, the value GREG is assigned to A\$(U-1) through the holding variable TEMP\$. This transaction looks like this:





# Worksheet 9.3

Part I Determine the output that should result from the following programs:

a) 10 OPEN 2,8,2,"NUMBERS,S,W"  
20 OPEN 3,8,3,"EXAMPLE,S,W"  
30 DEF FNA(X)=3\*X+1  
40 FOR I=1 TO 6  
50 IF FNA(I)/2=INT(FNA(I)/2) THEN PRINT#2,FNA(I):GOTO 70  
60 PRINT#3,FNA(I)  
70 NEXT I  
80 CLOSE 2: CLOSE 3  
90 OPEN 6,8,6,"EXAMPLE,S,R"  
100 INPUT#6,G  
110 PRINT FNA(G)  
120 IF (STATUS AND 64) <> 64 THEN 100  
130 CLOSE 6  
140 END

b) 10 OPEN 4,8,4,"FILE,S,W"                   110 OPEN 9,8,9,"FILE,S,R"  
20 FOR C=1 TO 4                               120 INPUT#9,S  
30 READ L,M,N                               130 INPUT#9,Z  
40 PRINT#4,L                                 140 PRINT S+Z  
50 PRINT#4,M                                 150 IF (STATUS AND 64) <> 64  
60 PRINT#4,N                                 THEN 120  
70 NEXT C                                    160 CLOSE 9  
80 CLOSE 4                                   170 END  
90 DATA 3,6,7,9,9,3  
100 DATA 5,7,4,6,1,2

Part II Find the errors (if any) in the following bubble sort programs and suggest corrections:

a) 10 FOR X=1 TO 4  
20 READ C\$(X)  
30 NEXT X  
40 FOR A=1 TO 3  
50 FOR B=4 TO A+1 STEP -1  
60 IF C\$(B)>C\$(B-1) THEN 90  
70 C\$(B)=C\$(B-1)  
80 C\$(B-1)=C\$(B)  
90 NEXT B  
100 NEXT A  
110 DATA EARTH,STARS,MOON,SUN  
120 END

b) 10 FOR A=1 TO 5  
20 READ H(A)  
30 NEXT A  
40 FOR T=1 TO 4  
50 FOR U=5 TO T+1 STEP -1  
60 IF H(T)>H(U) THEN 100  
70 T=H(U)  
80 H(U)=H(U-1)  
90 H(U-1) = T  
100 NEXT U  
110 NEXT T  
120 DATA 34,23,14,67,8  
130 END

# Worksheet 9.3 Solutions

Part I Output should be as follows:

```
a)  RUN
      22
      40
      58

      READY.
```

```
(b)  RUN
      9
      16
      12
      12
      10
      3

      READY.
```

Part II (a) If this program is to sort its list properly, a different "switching" method must be devised. The inclusion of a temporary holding value for A\$(B) must be inserted, perhaps at line 65, so that the two variables in question do not become equal to A\$(B-1). The following lines would correct the program:

```
65     TEMP$=A$(B)
70     A$(B)=A$(B-1)
80     A$(B-1)=TEMP$
```

(b) There are several errors in this program. First, line 60 should read

```
60     IF H(U-1) < H(U) THEN 100
```

Secondly, the outer loop variable, T, is used to store the temporary value in the switching process. This should be changed to another variable, Z, for example.

# Chapter 9 Supplementary Problems

- 1) A cassette file named SUSPECT was created by law-enforcement experts using the following program:

```
10 OPEN 1,1,1,"SUSPECT"  
20 FOR I=1 TO 25  
30   INPUT "SUSPECT'S NAME ";S$  
40   INPUT "EYE COLOR ";E$  
50   PRINT#1,S$;" ";E$  
60 NEXT I  
70 CLOSE 1  
80 END
```

Each record contains two fields. The first field is occupied by the suspect's name while the second contains his eye color. Write a program to select one suspect randomly, with brown eyes, to be questioned.

- 2) a) Write a program to load your friends' names and birthdays into a diskette file called BIRTHDAY. Allow each birthday to be entered in the form

MMDD

For example, September 5 would be represented in the file by

0905

b) Write a program which allows the user to input a month and will then print out the names of all the people on file born in that month. A sample run would appear as follows:

```
RUN  
MONTH (NUMERIC) ?9  
  
NAME      DAY  
----      ---  
  
JOHN      8  
MARY      13  
MICHAEL   17  
KYRA      24  
  
READY.
```

Note that the list is printed out in the order of the day of birth within the month. Use a bubble sort to sort the names by day of birth.

- 3) Write a program that will sort the list of states below in reverse alphabetical order using a bubble sort. Print the sorted list using the program fragment below for data.

```
900 DATA KANSAS,ARKANSAS,NEW YORK,WISCONSIN  
910 DATA NEW JERSEY,CALIFORNIA,OKLAHOMA,UTAH  
920 DATA SOUTH CAROLINA,MICHIGAN,TEXAS,OHIO
```

# Chapter 9 Supplementary Problem Solutions

- 1) 10 OPEN 1,1,0,"SUSPECT"  
20 I = INT(1+RND(0)\*25)  
30 FOR X=1 TO I  
40 INPUT#1,S\$,E\$  
50 NEXT X  
60 CLOSE 1  
60 IF E\$<>"BROWN" THEN PRINT "PLEASE REWIND THE TAPE  
TO THE BEGINNING OF THE FILE":INPUT "HIT RETURN WHEN  
READY ";D\$:GOTO 10  
70 PRINT "SUSPECT ";S\$;" WILL UNDERGO QUESTIONING."  
80 END
- 2a) 10 OPEN 2,8,2,"BIRTHDAY,S,W"  
20 INPUT "NAME,DATE ";N\$,B\$  
30 PRINT#2,N\$;" ";B\$  
40 INPUT "ANOTHER Y/N ";A\$  
50 IF A\$="Y" THEN 20  
60 CLOSE 2  
70 END
- b) 10 OPEN 9,8,9,"BIRTHDAY,S,R"  
20 INPUT "MONTH (NUMERIC) ";M  
30 INPUT#9,N\$,B\$  
40 IF VAL(LEFT\$(B,2)) = M THEN GOSUB 300  
50 IF (STATUS AND 64) <> 64 THEN 30  
60 REM START BUBBLE SORT  
70 FOR T=1 TO Q-1  
80 FOR U=Q TO T+1 STEP -1  
90 IF D(U) > D(U-1) THEN 120  
100 T\$=N\$(U):N\$(U)=N\$(U-1):N\$(U-1)=T\$  
110 Z=D(U):D(U)=D(U-1):D(U-1)=Z  
120 NEXT U  
130 NEXT T  
140 PRINT  
150 PRINT "NAME";TAB(20);"DAY"  
160 PRINT "-----";TAB(20);"----"  
170 PRINT  
180 FOR E=1 TO Q  
190 PRINT N\$(E);TAB(20);D(E)  
200 NEXT E  
210 CLOSE 9  
220 END  
300 Q = Q + 1  
310 N\$(Q)=N\$: D(Q)=VAL(RIGHT\$(B\$,2))  
320 RETURN
- 3) 10 DIM A\$(12)  
20 FOR X=1 TO 12  
30 READ A\$(X)  
40 NEXT X  
50 FOR T=1 TO 11  
60 FOR U=12 TO T+1 STEP -1  
70 IF A\$(U) < A\$(U-1) THEN 110



```
80     TEMP$ = A$(U)
90     A$(U) = A$(U-1)
100    A$(U-1) = TEMP$
110    NEXT U
120  NEXT T
130  FOR F=1 TO 12
140    PRINT A$(F)
150  NEXT F
900  DATA KANSAS,ARKANSAS,NEW YORK,WISCONSIN
910  DATA NEW JERSEY,CALIFORNIA,OKLAHOMA,UTAH
920  DATA SOUTH CAROLINA,MICHIGAN,TEXAS,OHIO
999  END
```



# Chapter 9 Test

Part I Determine the output that should result from each of the following programs:

a) 10 OPEN 4,8,4,"DATA1,S,W"  
20 READ Y  
30 R = R + Y  
40 IF R>0 THEN PRINT#4,Y:GOTO 60  
50 PRINT Y  
60 IF R<16 THEN 20  
70 DATA 3,-2,-4,-6,4,8,1,4,3,-4  
80 DATA 4,7,6,2,3,-8,5,6,7,8  
90 CLOSE 4  
100 END

b) After program (a) has been run  
  
10 OPEN 3,8,3,"DATA1,S,R"  
20 INPUT#3,J  
30 PRINT J  
40 IF (STATUS AND 64) <> 64 THEN 20  
50 CLOSE 3  
60 PRINT "FINISHED"  
70 END

c) 10 DIM N\$(15)  
20 N=15  
30 FOR X=1 TO N  
40 READ N\$(X)  
50 NEXT X  
60 FOR T=1 TO N-1  
70 FOR I=N TO T+1 STEP-1  
80 IF N\$(I)<N\$(I-1)  
THEN 100  
90 T\$=N\$(I):N\$(I)=N\$(  
I-1):N\$(I-1)=T\$  
100 NEXT I  
110 NEXT T  
120 FOR X=N TO 1 STEP -3  
130 PRINT N\$(X-1)  
140 NEXT X  
150 DATA CAT,DOG,APE,AXE  
160 DATA HAT,HIM,HER,MOM  
170 DATA DAD,RED,ZAP,FOG  
180 DATA HOT,WET,DRY  
190 END

Part II Circle the programming error(s) within each program given. Explain what must be done to correct the error. If no error exists, state so.

a) 10 OPEN 3,1,1,"DOLLAR,S,W"  
20 FOR X=1 TO 4  
30 READ A,B,C  
40 IF A\*.05+B\*.1+C\*.25=1 THEN 60  
50 PRINT#1,A,B,C  
60 NEXT C  
70 CLOSE 1  
80 DATA 2,4,2,1,0,3,  
90 DATA 3,1,2,4,2,3  
100 END

b) 10 OPEN 1,8,1,"FILE2,S,W"  
20 N = 5  
30 FOR Y=1 TO N  
40 READ X\$(N)  
50 PRINT#1,X\$(N)  
60 NEXT N  
70 CLOSE 1  
80 END

c) 10 R=10  
20 FOR X=1 TO 11  
30 READ N\$(X)  
40 NEXT X  
50 FOR T=1 TO R-1  
60 FOR I=R TO T+1 STEP -1  
70 IF N\$(I)>N\$(I-1)  
THEN 90  
80 N\$(I)=N\$(I-1):N\$(  
I-1)=N\$(I)  
90 NEXT I  
100 NEXT T  
110 FOR X=1 TO 11  
120 PRINT N\$(X)  
130 NEXT X  
140 DATA MIKE,CHRIS,GREG  
150 DATA JOHN,FRED,BRUCE  
160 DATA CAROL,JON,EILEEN  
170 DATA JOE,LANCE  
180 END

Part III Write the following programs as concisely as possible:

- a) A file named HORSE exists on a diskette. Each record contains the name of a horse and the number of records in the file is unknown. Write a program to print out the number of horses in the file. Do not allow an error message to occur.
  
- b)
  - i) Write a program that will create a file called WEEKEND on a diskette. Allow the user, a busy executive, to enter 10 popular, relaxing weekend activities (such as GOLFING, FLYING or COMPUTING).
  
  - ii) Write a program which will alphabetize the list of activities created in program (i). Store this list in a file called FUN.
  
  - iii) Finally, write a program which allows the user to ask for an idea for the weekend. Have the computer randomly choose an activity using the file FUN and print it out.

# Chapter 9 Test Solutions

Part I [ 8@ = 24 ]

- |    |          |    |        |
|----|----------|----|--------|
| a) | RUN      | c) | RUN    |
|    | -4       |    | AXE    |
|    | -6       |    | DOG    |
|    | 4        |    | HAT    |
|    |          |    | HOT    |
|    | READY.   |    | WET    |
| b) | RUN      |    | READY. |
|    | 3        |    |        |
|    | -2       |    |        |
|    | 8        |    |        |
|    | 1        |    |        |
|    | 4        |    |        |
|    | 3        |    |        |
|    | -4       |    |        |
|    | 4        |    |        |
|    | 7        |    |        |
|    | FINISHED |    |        |
|    | READY.   |    |        |

Part II [ 8@ = 24 ]

a) Line 10 should read `10 OPEN 3,1,1,"DOLLAR"`. This is the proper OPEN format for a cassette file. Line 50 should read `50 PRINT#3,A;"",B;"",C`. Line 70 should close the file number, 3, in this case; it should read `70 CLOSE 3`. Finally, the comma at the end of the data in line 80 should be deleted.

b) Line 10 attempts to OPEN a diskette file on channel 1, which is reserved for the Operating System of the computer. It would be correct to open the diskette file on any channel number between 2 and 14. If you chose 2, the line would read

```
10 OPEN 2,8,2,"FILE2,S,W"
```

In lines 40 and 50 the subscripted variable should be the loop variable, Y. Also, line 60 should read `60 NEXT X`. To be consistent with earlier examples, lines 50 and 70 should use the new channel number when dealing with the file. Line 50 should read `50 PRINT#2,X$(Y)` and line 70 should read `70 CLOSE 2`, if we are to be consistent with line 10. Finally, there is no DATA for the READ to read.

c) Line 10 should read `10 R=11`. In line 60 the `STEP -1` is missing from the `FOR..NEXT` line. The switching process in line 80 does not use a temporary storing value. Finally, since the array of N\$ has 11 members, it must be DIMensioned at the beginning of the program.

## Part III

```

a) 10 OPEN 3,8,3,"HORSE,S,R" [ 10 ]
    20 INPUT#3,H$
    30 N=N+1
    40 IF (STATUS AND 64)<>64 THEN 20
    50 PRINT "THERE ARE ";N;" HORSES IN THE FILE."
    60 CLOSE 3
    70 END

b) [ 10 ]
  i) 10 OPEN 7,8,7,"WEEKEND,S,W"
      20 FOR X=1 TO 10
      30   INPUT "ACTIVITY ==>> ";A$
      40   PRINT#7,A$
      50 NEXT X
      60 CLOSE 7
      70 END

  ii) 10 OPEN 7,8,7,"WEEKEND,S,R" [ 22 ]
       20 FOR X=1 TO 10
       30   INPUT#7,A$(X)
       40 NEXT X
       50 CLOSE 7
       60 FOR B=1 TO 9
       70   FOR U=10 TO B+1 STEP -1
       80     IF A$(U) > A$(U-1) THEN 120
       90     TEMP$=A$(U)
      100     A$(U)=A$(U-1)
      110     A$(U-1)=TEMP$
      120   NEXT U
      130 NEXT B
      140 OPEN 5,8,5,"FUN,S,W"
      150 FOR I=1 TO 10
      160   PRINT#5,A$(I)
      170 NEXT I
      180 CLOSE 5
      190 END

  iii) 10 OPEN 4,8,4,"FUN,S,R" [ 10 ]
        20 V=INT(10*RND(0)+1)
        30 FOR X=1 TO V
        40   INPUT#4,A$
        50 NEXT X
        60 PRINT "WHY DON'T YOU GO ";A$;" THIS WEEKEND?"
        70 CLOSE 4
        80 END

```

# Final Examination

## Part I Multiple Choice

Determine the best response and place the corresponding letter in the blank.

\_\_\_ 1) The statement:

```
X = PEEK(211)
```

- a) Places the value 211 into memory location 211.
- b) Erases the contents of memory location 211.
- c) Assigns the value stored in memory location 211 to the variable X.
- d) Assigns the value of X to memory location 211.
- e) Is an illegal statement.

\_\_\_ 2) In the statement:

```
DEF FN Z(X) = 3 * X + 5
```

- a) If X = 4 then the variable will equal 17.
- b) If J = 5 then FN Z(J) will equal 20.
- c) Another variable may not be substituted for X when the function is evaluated.
- d) If T = 12 then FN Z(T) will equal 36.
- e) X is the name of the function.

\_\_\_ 3) What is the result of the following:

```
10 FOR I = 1 TO 2000
20   PRINT " ";
30 NEXT I
40 PRINT
```

- a) no noticeable output
- b) prints blank spaces down the side of the screen
- c) prints 2000 blank spaces
- d) puts the user into reverse mode
- e) prints 2000 reverse mode blank spaces

\_\_\_ 4) What is the decimal equivalent of the binary number 101?

- a) 2
- b) 3
- c) 4
- d) 5
- e) 6

\_\_\_ 5) Determine the output of the following program:

```
10 P$ = "AB"
20 PRINT CHR$(ASC(P$))
```

- a) AB
- b) A
- c) 65
- d) 66
- e) ERROR MESSAGE

- \_\_\_ 6) How many bits constitute a byte?
- a) 65537            c) 8                    e) 2  
b) 16                d) 256
- \_\_\_ 7) What is the largest positive value an integer variable may have in the computer?
- a) 32767                    c) 65537            e) NO LIMIT  
b) 256                      d) 3E39
- \_\_\_ 8) Determine the output of the following program:
- ```
10 N$ = "11ELEVEN"
20 PRINT MID$(N$,VAL(N$)-6,LEN(N$)-5)
```
- a) EVE                    c) ELEVEN                    e) EVEN  
b) ELE                    d) 11
- \_\_\_ 9) An unwanted file named "WORK" is removed from a diskette by using the command(s):
- a) ERASE "WORK",8                    d) OPEN 15,8,15  
b) OPEN 15,8,15                      PRINT#15,"S:WORK"  
   PRINT#15,"N:WORK,DELETE"        CLOSE 15  
   CLOSE 15                            e) LOAD "WORK",8  
c) DELETE "WORK",8                    NEW
- \_\_\_ 10) Consider the following program:
- ```
10 OPEN 2,8,2,"WORK,S,W"
20 FOR I = 1 TO 5
30 INPUT N$,A
40 PRINT#2,N$;" ";A
50 NEXT I
80 CLOSE 2
```
- How many records are put into the file "FILE10" when the program is run?
- a) 0                    c) 10                    e) 20  
b) 5                    d) 13
- \_\_\_ 11) Which of the following statements are true:
- a) The computer can access a cassette file faster than a diskette file.  
b) Both cassettes and diskettes have built in directories which can be listed.  
c) The computer can access a diskette file faster than a cassette file.  
d) Neither cassettes nor diskettes will store programs.  
e) Once a file is accessed on either the cassette or diskette, it is automatically erased.



- \_\_\_ 12) Which of the following functions does NOT have a string as its argument?
- a) LEFT\$                      c) RIGHT\$                      e) STR\$  
b) LEN                              d) MID\$
- \_\_\_ 13) Which of the following bytes represents the decimal number 45?
- a) 0 0 0 1 1 1 0 1  
b) 0 0 1 0 0 1 1 1  
c) 0 0 1 0 1 0 1 1  
d) 0 0 0 1 1 1 0 1  
e) 0 0 1 0 1 1 0 1
- \_\_\_ 14) Which of the following is a legal operation on the computer?
- a) SQR(-4)                      c) SIN(0)                      e) LOG(-1)  
b) TAN( $\pi/2$ )                      d) EXP(100)
- \_\_\_ 15) To convert an angle from degrees to radians
- a) Multiply by 180 then divide by PI  
b) Multiply by then product of 180 and PI  
c) Divide by the product of 180 and PI  
d) Multiply by PI then divide by 180  
e) Multiply by PI then divide by 360

Part II Exact output

Determine the output of each of the following programs:

- 1) 

```
10 DEF FN T(X) = X * 2 + 5
20 Y = SGN(8/4-4)
30 X = ABS(-2↑3)
40 Z = FN T(X) - FN T(Y)
50 PRINT Y:PRINT X:PRINT Z
```
  
- 2) 

```
10 A$ = "MISSISSIPPI"
20 L = LEN(A$)
30 FOR I = 1 TO L - 1
40     IF MID$(A$,I,1) = MID$(A$,I+1,1) THEN
        PRINT MID$(A$,I,2)
50 NEXT I
```
  
- 3) 

```
10 REM NOTE THAT ASC("A") = 65
20 PRINT ASC("B")
30 FOR I = 1 TO 6
40     READ X
50     PRINT CHR$(X);
60 NEXT I
70 DATA 83,85,77,77,69,82
```

Part III Locate the errors

Circle the programming error(s) within each program given. Explain what must be done to correct the error. If no error exists, state so.

```
1)  10  REM PRINT TEN WORDS BACKWARDS
     20  FOR Y = 1 TO 10
     30    INPUT W$
     40    FOR X = LEN$(W$) TO 1
     50      PRINT MID(W$,1,X),
     60    NEXT Y
     70  NEXT X
     80  END
```

```
2)  10  OPEN 0,8,3,"COLORS,S,W"
     20  INPUT#1,"WHAT COLOR";C$
     30  PRINT#0,C$
     40  INPUT "ANOTHER Y/N";N$
     50  IF N$ = Y THEN 20
     60  CLOSE 3
     70  END
```

## Part IV Programs

Write each of the following programs as concisely as possible. Good programming style is important.

- 1) A company has twenty-five employees with employee numbers ranging from 1 to 25. Write a program which will create a random access file named EMPLOYEE and allow the user to enter the list of employees. File record numbers will correspond to employee numbers. A schematic follows:

NAME\$	N
BLECKMAN	0
SANDERSORT	0
BLITHENSPUTH	0
.	.
:	:
.	.

The N column corresponds to "CREDIT POINTS" given by the boss. When the file is created, this column contains zeroes. The employee with the longest name is "Blithensputh". Note that no one has ever gotten more than three or four credit points (i.e., single digits).

- 2) Write a program that will access the file EMPLOYEE created in problem 1 and allow the boss to pick one of the following three options:
  - a) Store the name of a new employee in the file and set N to 0.
  - b) Input the name of an employee who has done an especially good job and add one point to his or her credit, N.
  - c) List all employees with credit equal to or above a number entered by the boss.

Each of the three options should be programmed in its own separate subroutines. Programming style is important.

# Final Examination Solutions

The credit for each problem is given in brackets [ ].

Part I [ 2@ = 30 ]

- |    |   |     |   |     |   |
|----|---|-----|---|-----|---|
| 1) | D | 6)  | C | 11) | C |
| 2) | B | 7)  | A | 12) | E |
| 3) | A | 8)  | A | 13) | E |
| 4) | D | 9)  | D | 14) | C |
| 5) | B | 10) | B | 15) | D |

Part II [ 5@ = 15 ]

- |    |    |    |    |    |        |
|----|----|----|----|----|--------|
| 1) | -1 | 2) | SS | 3) | 66     |
|    | 8  |    | SS |    | SUMMER |
|    | 18 |    | PP |    |        |

Part III [ 5@ = 15 ]

- Line 40 should be FOR X = LEN(W\$) TO 1 STEP -1
  - Line 50 should be PRINT MID\$(W\$,X,1)
  - Line 60 should be NEXT X
  - Line 70 should be NEXT Y
- Line 10 should be OPEN 2,8,3,"COLORS,S,W"
  - Line 20 should be INPUT "WHAT COLOR";C\$
  - Line 30 should be PRINT#2C\$
  - Line 50 should be IF X\$ = "Y" THEN 20
  - Line 60 should be CLOSE 2

## Part IV

```

1)  10 OPEN 2,8,2,"EMPLOYEE,S,W"                [ 20 ]
    20 N=0
    30 INPUT "ENTER EMPLOYEE NUM,NAME";E,N$
    40 PRINT#2,E;" ";N$;" ";N
    50 INPUT "ANOTHER Y/N";X$
    60 IF X$="Y" THEN 40
    70 CLOSE2
    80 END

2)  10 PRINT "[1] NEW EMPLOYEE"                [ 30 ]
    20 PRINT "[2] ADD CREDIT"
    30 PRINT "[3] LIST OUT"
    40 INPUT X
    50 ON X GOSUB 1000,2000,3000
    60 END
    1000 REM * ADD NEW EMPLOYEE *
    1010 OPEN 2,8,2,"EMPLOYEE,S,A"
    1020 INPUT "ENTER EMPLOYEE NUM,NAME";E,N$
    1030 PRINT#2,E;" ";N$;" ";N;0
    1040 CLOSE 2
    1050 RETURN
    2000 REM * ADD CREDIT *
    2010 OPEN 2,8,2,"EMPLOYEE,S,R":OPEN 3,8,3,"TEMP,S,W"
    2020 INPUT "ENTER NAME";T$
    2030 INPUT#2,E,N$,N
    2040 IF T$=N$ THEN N=N+1
    2050 PRINT#3,E;" ";N$;" ";N
    2060 N=0
    2070 CLOSE 2: CLOSE 3
    2080 OPEN 15,8,15
    2090 PRINT#15,"S:EMPLOYEE"
    2100 PRINT#15,"R:EMPLOYEE=TEMP"
    2110 CLOSE 15: RETURN
    3000 REM * LIST OUT *
    3010 OPEN 2,8,2,"EMPLOYEE,S,R"
    3020 INPUT "ENTER # OF POINTS";P
    3030 INPUT#2,E,N$,N
    3040 IF (STATUS AND 64) = 64 THEN CLOSE 2: RETURN
    3050 IF N<P THEN 330
    3060 PRINT E,N$,N
    3070 GOTO 3030

```

