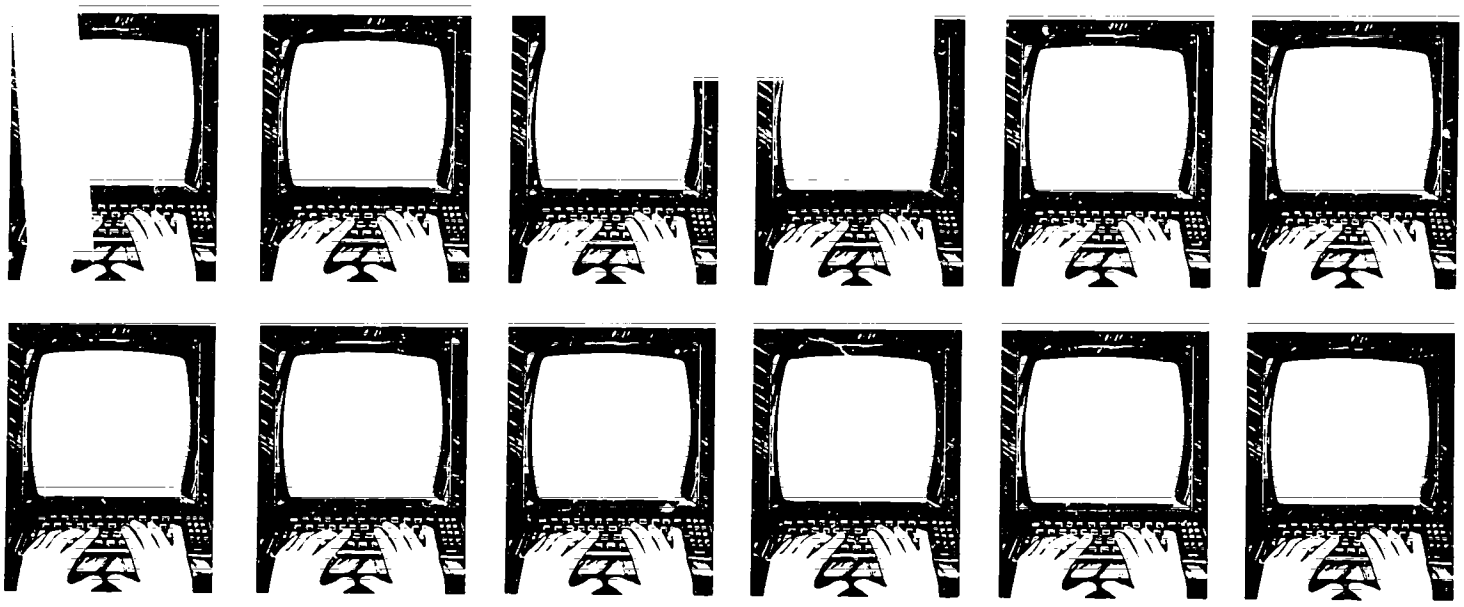
DOCUMENT RESUME

ED 280 432 IR 012 568

TITLE	Basic Programming II: Course Guide. Revised Edition.
INSTITUTION	Hawaii State Dept. of Education, Honolulu. Office of Instructional Services.
REPORT NO	RS-86-9863
PUB DATE	Jun 86
NOTE	205p.
PUB TYPE	Guides - Classroom Use - Guides (For Teachers) (052) -- Computer Programs (101) -- Reports - Descriptive (141)
EDRS PRICE	MF01/PC09 Plus Postage.
DESCRIPTORS	Business Education; Classification; Computer Graphics; *Computer Literacy; Computer Science Education; Course Descriptions; *Course Objectives; Ethics; *Learning Activities; Mathematics Instruction; *Microcomputers; *Programing; Programing Languages; Resource Materials; Secondary Education; State Curriculum Guides; *Teaching Methods
IDENTIFIERS	*BASIC Programing Language

ABSTRACT

This guide is designed to provide teachers with guidelines and suggested activities for teaching a one-semester advanced programming course--BASIC Programming II--for the ninth through twelfth grades. Although primarily oriented toward mathematics, the guide does offer sample applications in business that also address the needs of students with a variety of academic backgrounds. Intended to serve as a framework of goals and activities upon which the teacher can organize, build, and expand his or her course, the guide provides a course description, course requirements, a course outline, a syllabus, course management considerations, sample activities and programs, and suggested resources. The activities include teaching strategies for introducing concepts, developing specific skills, or reinforcing previously learned aspects of BASIC. Topics covered include a Review of Hardware/Software Considerations, Structured Programming via Subroutines, Subscripted Variables and Problem-Solving Strategies, BASIC Functions and Graphics, File Handling and Term Project, Data Structures, and Computer Ethics and Impact on Society. A taxonomy of goals, objectives, and student expectations is appended, as well as samples of forms for use with the course, a description of a motivation technique, ASCII codes, and lists of recommended textbooks, teaching aids and references, software, and audiovisual materials. (DJR)



BASIC PROGRAMMING II
COURSE GUIDE

Office of Instructional Services/General Education Branch
Department of Education □ State of Hawaii □ RS 86-9863 (Revision of RS 85-8913) □ June 1986

The Honorable George R. Ariyoshi
Governor, State of Hawaii

BOARD OF EDUCATION

Randal Yoshida, Chairperson
Sherwood M. Hara, First Vice-Chairperson
Charles Norwood, Second Vice-Chairperson

Rev. Darrow L.K. Aiona	Ronald Nakano
Margaret K. Apo	John R. Penebacker
Mako Araki	Akira Sakima
Dr. Hatsuko F. Kawahara	Meyer M. Ueoka
Michael Matsuda	William K. Waters

Francis M. Hatanaka, Superintendent of Education
Dr. Margaret Y. Oda, Deputy Superintendent

Bartholomew A. Kane, State Librarian

Claudia Chun, Assistant Superintendent
Office of Instructional Services

Vernon H. Honda, Assistant Superintendent
Office of Business Services

Albert Yoshii, Assistant Superintendent
Office of Personnel Services

William Araki, District Superintendent
Leeward District Office

Gordon Kuwada, District Superintendent
Central District Office

Lokelani Lindsey, District Superintendent
Maui District Office

Dr. Kiyoto Mizuba, District Superintendent
Hawaii District Office

Dr. Mitsugi Nakashima, District Superintendent
Kauai District Office

Claudio Suyat, District Superintendent
Honolulu District Office

Kengo Takata, District Superintendent
Windward District Office

11/85

FOREWORD

The intent of this course guide is to provide teachers with guidelines and suggested activities for teaching an advanced programming course in the BASIC language. The guide supports a one-semester course, BASIC Programming II.

Although primarily oriented toward mathematics, this document does offer sample applications in business. The sample applications also address the needs of students with a variety of academic backgrounds.

The revision of the original draft of this guide is based on recommendations from teachers and other specialists during the pilot study conducted in the 1985-86 school year.

It is hoped that this guide will prove to be useful to those secondary teachers who wish to implement BASIC Programming II at their schools.

Francis M. Hatanaka
Superintendent

ACKNOWLEDGMENT

Development of the original draft for the BASIC Programming II Course Guide, evaluation of the pilot use of the draft and subsequent revision of the document were conducted by Developing Instructional Computing, an ECIA, Chapter 2 Project, under the management of Rosemary Hill Darabian, General Education Branch, Office of Instructional Services.

Appreciation is extended to the teachers in the pilot schools whose input was most helpful in the revision of the guide. They are:

Charles Adams	McKinley High School
Naomi Nishida	Waipahu High School
Sandra Shirachi	Castle High School
Fay Zenigami	Nanakuli High School

Recognition for invaluable input is also extended to two teachers who teach BASIC in business education.

Amy Fujii	Kaimuki High School
Linda Kishimoto	McKinley High School

The Department of Education also acknowledges the following state personnel who provided recommendations and support in the development of the guide.

Evelyn Horiuchi	Educational Specialist, Computer Education
Kathleen Nishimura	Educational Specialist, Mathematics
Yukio Toyama	Educational Specialist, Business Education

Finally recognition is extended to those teachers in the summer in-service training session who provided vital feedback on the strengths and weaknesses of the original draft.

BASIC PROGRAMMING II
TABLE OF CONTENTS

BASIC PROGRAMMING II
TABLE OF CONTENTS

INTRODUCTION

This guide for BASIC Programming II is not intended to serve as a textbook but rather as a framework of goals and activities, upon which the high school teacher can organize, build and expand his or her course.

Although there is an abundance of resource materials and textbooks available for an introductory course in BASIC, there are considerably fewer materials for an advanced course in BASIC at the secondary school level. Many of the college texts are "cut and dry" conveyers of facts, format and syntax, with little emphasis on areas such as problem solving and illustrating effective versus less efficient programming styles. The latter chapters of most introductory high school texts do offer material appropriate for this course, but no one text is recommended, as none provides a complete coverage of all topics within the course objectives. Therefore, a combination of texts and reference books should be available to the teacher of BASIC Programming II. Refer to the lists of Recommended Textbooks, Teaching Aids, Software and References in the Resources section of this guide.

Because students enrolled in this course will have successfully completed BASIC Programming I or the equivalent, their background in the subject matter should be quite similar. Whether or not students have successfully completed Algebra I, they should have a solid understanding of the concept "variable." In many cases the same teacher will be teaching both programming courses, enabling the students to progress smoothly from one course to the next. However, regardless of who teaches BASIC Programming II or its equivalent (such as in the Mathematics or Business Departments), a short period of overlap between the two courses should prove valuable in reviewing key concepts from the past and projecting their potential uses for the future.

COURSE DESCRIPTION

This course is designed to reinforce and extend the programming skills learned in BASIC Programming I and to develop the following fundamental concepts and skills of programming, using the computer language BASIC:

- write logically structured, well-documented programs
- select and use appropriate algorithms
- design and use numeric and string arrays and matrices
- design appropriate error trapping routines
- design and manipulate sequential and random access files
- distinguish types of data structures
- recognize the ethical and social implications of computer use.

GRADES: 9-12

GOALS:

The four goals for Computer Science are:

1. The student will demonstrate competence in using computers;
2. The student will use the computer as a tool for problem solving and decision making;
3. The student will recognize the impact of computers in daily life;
4. The student will investigate educational and career opportunities in computer-related professions.

The taxonomy of goals, objectives and student expectations is derived from the Computer Science component of the Computer Literacy Framework. The recently revised version of this taxonomy can be found in the Appendix.

The taxonomy, which is an analytical outline of the Framework, has these design features:

1. Goals are listed and subdivided into objectives which are further subdivided into student expectations;
2. Objectives are phrased so that they can be used to identify relevant classroom materials;
3. The numerical identification system is designed to accept expansion or reduction of goals and objectives as experience requires.

STUDENT EXPECTATIONS:

The numeric system of the Taxonomy consists of one-, two- and three-digit numbers, each separated by decimal points. The first digit always represents a goal, whether it stands by itself or in a two- or three-digit number. Likewise, the second digit represents an objective and the third digit stands for a student expectation. The twenty-nine student expectations for Computer Science are delineated below under the appropriate goals.

GOAL 1 The student will demonstrate competence in using computers.

1.1.1. The student selects and uses appropriate resources (manuals, programs, peripherals, etc.) for performing a task.
1.1.2. The student adapts programs to solve specific problems.
1.1.3. The student evaluates and compares computer programs (prepackaged and student's own).
1.2.1. The student demonstrates through a project the processing of information.

1.2.2. The student implements routines to process information through searching, sorting, deleting, updating, summarizing, storing, etc.

1.2.3. The student explains the major parts and functions of a computer system (e.g., CPU - registers, accumulators; memory - addressing; peripherals - cylinders, tracks, sectors).

1.3.1. The student demonstrates the ability to clearly define problems and to subdivide a particular problem into logical subproblems.

1.3.2. The student designs structured solutions to problems - algorithms - by applying the principles of top-down design methodology.

1.3.3. The student properly implements the available control structures - sequence, iteration and branching - when coding algorithms into a specific high-level language.

1.3.4. The student designs and uses numeric and string arrays and matrices.

1.3.5. The student demonstrates the ability to anticipate, identify, isolate and correct errors.

1.3.6. The student enhances the readability and clarity of his or her program by including appropriate documentation.

GOAL 2 The student will use the computer as a tool for problem solving and decision making.

2.1.1. The student describes the major types of data structures available to the high-level language being studied and understands their uses and limitations.

2.1.2. The student recognizes and appropriately utilizes elementary data structures in solving problems.

2.2.1. The student recognizes and appropriately utilizes elementary algorithms in solving problems.

2.2.2. The student designs and implements his or her own algorithms in solving some types of programming problems.

2.3.1. The student creates and utilizes sequential data files for file-processing programs.

2.3.2. The student creates and utilizes random data files for file-processing programs.

2.3.3. The student easily uses mathematical and string manipulation functions specific to the high-level language being studied.

2.3.4. The student designs a variety of graphics programs.

2.3.5. The student experiences working as a team in a programming environment which simulates the actual field where each team is responsible for developing one module in a larger programming system.

2.4.1. The student recognizes and uses computer application tools.

2.4.2. The student values efficient information processing.

GOAL 3 The student will recognize the impact of computers in daily life.

3.1.1. The student identifies computer applications in business, industry, scientific research, medicine, government, education, health and social services, recreation, creative arts, etc.

3.1.2. The student appreciates the economic benefits of computerization for society.

3.1.3. The student understands that computers can be used to effect distribution and use of economic and political power, and used in criminal and other anti-social activities that affect society in undesirable ways.

3.2.1. The student accepts responsibility for following school and lab rules pertaining to computer ethics.

GOAL 4 The student will investigate educational and career opportunities in computer-related professions.

4.1.1. The student identifies careers that involve computers directly (support service, technical and scientific careers, data management, programming analysis, etc.)

4.1.2. The student compares educational requirements and opportunities for careers that involve computers.

PREREQUISITES:

BASIC Programming I or the equivalent.

MATERIALS:

Microcomputers with at most two students per computer per class; textbooks and other reference books on the programming language BASIC.

TIME FOR ACTIVITIES:

Time will vary from one day to one month for the programming assignments, depending on the type of assignment (an exercise, a practice program or a term project), the number of computers available, the number of students in the class(es) and the number of hours the lab is available.

TEACHER PREPARATION:

The teacher should be a programmer in BASIC, i.e., know enough about the language to write programs, using features of the language listed in the Course Outline, following this section. Knowledge of data structures and of manipulating sequential and random access files is essential.

COURSE OUTLINE

I. Computer Memory and Processing

- A. Disk Operating System
- B. Compilers versus Interpreters
- C. Bits, Bytes and Binary System
- D. Binary Code versus ASCII Code
- E. Data Types

II. Programming Fundamentals

- A. Problem Solving
- B. Top-Down Design
- C. Subroutines or Subprograms
- D. Data Entry and Error Checking Subroutines
- E. Documentation
- F. Debugging Techniques

III. Functions in BASIC

- A. Intrinsic Functions
  - 1. Mathematical Functions
  - 2. String Manipulation Functions
- B. User-Defined Functions

IV. Lists and Tables in BASIC

- A. One-Dimensional Arrays
- B. Two-Dimensional Arrays
- C. Searching and Sorting Arrays

V. Data Files

- A. Sequential File Processing
- B. Random File Processing

VI. Data Structures

- A. The Stack
- B. Queues and Linked Lists
- C. Trees

VII. Graphics

- A. Low-Resolution Graphics
- B. High-Resolution Graphics

VIII. Computer Ethics and Impact on Society

A. Computer Careers
B. Computer Uses and Misuses

SYLLABUS

I. Computer Hardware/Software Considerations

A. Computer Lab Rules

B. Proper Computer Care

C. Disk Operating System

D. Compilers versus Interpreters

E. Bits, Bytes and Binary System

1. Binary and ASCII Codes
2. Data Types

(Topics II and III should be emphasized throughout the course.)

II. Structured Programming

A. Control Structures

1. Sequence
2. Selection
3. Iteration or Repetition

B. Top-Down Design

1. Modular Programming
2. Limited Use of GOTO

C. Subroutines or Subprograms

III. Problem-Solving Strategies

A. Problem Definition and Flowchart or Algorithm Design

B. Coding and Error Checking Subroutines

C. Testing and Debugging Techniques

D. Documentation

(Topics IV and V could be interchanged or combined.)

IV. Subscripted Variables

A. One-dimensional Arrays

1. Searching Lists
2. Sorting Lists

B. Two-dimensional Arrays

1. DIM Statements
2. MAT-READ Statements (optional)

V. BASIC Functions

A. Intrinsic Functions

1. Mathematical Functions
   INT, RND, SQR, SGN, ABS, SIN, COS, TAN, LOG, EXP
2. String Manipulation Functions
   CHR$, ASC, INKEY$ or GET A$, LEFT$, MID$, RIGHT$, STR$, VAL, LEN

B. User-Defined Functions: DEF FN

VI. File Handling

A. Sequential Files

1. Statements and Functions for File Processing
2. File Commands

B. Random or Direct Files

1. Records and Fields
2. Random File Processing

VII. Graphics

A. Low- and High-Resolution Graphics

1. Plotting
2. Drawing

B. The Uses of Computer Graphics

VIII. Data Structures

A. The Stack

1. Definition and Function
2. Examples and Applications

B. Queues and Linked Li ts

1. Definitions and Functions
2. Examples and Applications

C. Trees

1. Definitions and Functions
2. Examples and Applications

(Topic IX should be addressed throughout the course.)

IX. Computer Ethics and Impact on Society

A. Implications

1. Computer Careers and Other Job Opportunities
2. Applications in Industry, Medicine, Data Processing, etc.

B. Issues

1. Computer Misuses and Crime
2. Unemployment and Depersonalization
3. Privacy

X. Assignments

A. Daily or Weekly Exercises

1. Class
2. Team
3. Individual

B. Weekly or Biweekly Short Programs

1. Team
2. Individual

C. Term Project

1. Pair of students
2. Individual

D. Unit Tests

REFERENCE MATERIALS

It is highly recommended that reference books be available which include the features unique to the computer being used and that programming assignments be designed to include some of these features whenever possible. The following is a list of references to consider:

1) Dictionary of computer terms;
2) Books or workbooks specifically written for the computer being used;
3) Periodicals;
4) Quick Reference Cards for the computer being used.

DISKETTES

Each student should have his or her own diskette for storing the assigned exercises and programs. Backups of his or her work should be made regularly by the student on a second diskette to ensure protection from loss due to disk damage. At least one of the diskettes should be kept in a secured file located in the lab or the classroom.

CLASSROOM MANAGEMENT

A lab setting with one student per computer is ideal. No more than two students working at a computer is recommended. In most cases there will be more students per class than microcomputers. Therefore, class time should be equally divided into work sessions to enable each individual student his or her turn. A system should be encouraged whereby students must first plan their project or assignment on paper before using the computers. Development of flowcharts or algorithms is an appropriate pre-programming activity. Other desk assignments or worksheets should be provided to involve all students in constructive activity while waiting for their computer time.

Sign-up sheets should be available for computer time outside of the class period. There should be a limit as to how many times a student signs up during a week. The student should cross off his or her name, when unable to keep the appointment, allowing other students a chance for that time slot. Penalties could be given to those who abuse the sign-up system. Refer to the sample sign-up sheet located in the Appendix of this guide.

With a computer lab setting, which is recommended, the teachers may need an assistant or two to maintain order in the classroom while he or she is in the lab or to help out in the lab while he or she remains in the classroom. Rewards for these assistants should include bonus points, extra computer time or the additional knowledge and experience gained by this opportunity.

ASSIGNMENTS AND GRADING

Assignments can include daily or weekly exercises on specific programming skills or concepts, weekly or biweekly short programs involving related course content and one or two term projects of a longer program incorporating most major concepts presented that quarter or semester. Programming assignments can be distributed one week before the due date of the previous program, and new material needed can be covered from that week. After the deadline has passed for any programming assignment, it is helpful to compare and discuss several student approaches at solving the problem. When there are no lectures or lab exercises, students can work on their flowcharts at their desks or enter their programs at the computers. A constant overlap of programming assignments during the semester enables maximum use of programming time and allows students who complete their programs early to have another project to prepare.

Adequate documentation is just as important as solving the problem. Documentation should be included as the program is written. This approach makes the debugging process easier for the student and the teacher. Students can work on their documentation at their desks before they enter the REM statements in their programs.

Some students may want to work in pairs on their programs. This generally works out well when both students have the same ability level and the work can be divided satisfactorily between the two. Both students should receive the same grade for the program. There are times when exercises are best done by the class as a whole, a team or pair of students or the individual. Students can learn from one another in group work, but they can also take advantage of the situation by letting others carry the load. It is left to the discretion of the teacher as to when students should work in groups or alone.

One suggestion for handling a class with a wide range of student proficiency is to assign several versions of a programming problem adjusted to an appropriate level of difficulty. The more skilled individuals can be challenged with the most difficult version, while the least able can be given a version that will help them grow without overwhelming them.

Concerning due dates for assignments, alternatives include giving extra points for the early completion of a program or not accepting late ones. However, there should be a ceiling on the number of extra points given, such as one point for each day an assignment is early, up to a maximum of five points. As suggested in the guide for BASIC Programming I, an earlier due date for the flowchart tends to motivate the student to start developing his or her program sooner than if both the flowchart and the program are due at the same time. This also allows the teacher adequate time to spot problems that a student may be having with the assignment.

A possible weight distribution for grading programming assignments follows:

30% flowchart and/or algorithm
30% translation of flowchart and/or algorithm into structured BASIC

20% documentation
15% valid output
5% other (such as adhering to the lab rules).

A sample of a grading sheet for programs is located in the Appendix.

SAMPLE ACTIVITIES AND PROGRAMS

INTRODUCTION TO ACTIVITIES

The suggested sample activity clusters and programming assignments on the following pages support the fundamental concepts and skills delineated in the Course Description, the student expectations enumerated in the Course Requirements and the content prescribed in the Course Outline. These activities provide teaching strategies for introducing certain concepts, developing specific skills or reinforcing previously learned aspects of BASIC. The syllabus included in this guide offers only a suggested order of content. It should be noted that within the sample clusters of activities, syllabus topics may not be in the exact order given, as some assignments can easily incorporate more than one topic. For instance, graphics may be integrated throughout the course, not presented just toward the end of the semester. The level of experience and skills of the students will most likely determine the order of presentation of topics and the degree to which they are covered.

Examples of most programs are geared for Apple equipment but can be modified for other brands of microcomputers.

The sample activities and programs are not intended to be the only method of presentation but are instead "starting points" from which teachers can expand, using their own approaches, ideas and creativity.

SAMPLE ACTIVITY CLUSTER #1

Review of Hardware/Software Considerations

Topics:

1. Computer Lab Rules
2. Proper Computer Care
3. Disk Operating System
4. Compilers versus Interpreters
5. Bits, Bytes and Binary System

Classroom Management:

The class as a group will be involved with these activities.

Materials:

Overhead projector and prepared transparencies;
Videotape player and the videotape, "Hardware and Software";
Demonstration microcomputer and large-screen monitor;
Handouts.

Time for Activities:

Approximately two to five days

Teacher Preparation and Procedures:

Prepare transparencies, handouts and the demo disk of BASIC and Pascal programs. All sample materials for these activities are located in the Sample Assignments and Materials for Activity Cluster #1.

1. Using the overhead projector, review the basic rules on lab behavior and proper computer care. (Emphasize any changes since last semester's course in BASIC Programming I.) Discuss any student concerns. Handout #1 - Computer Lab Rules is a sample set of rules that could be used.

2. Review the role of the Disk Operating System (DOS) or the CP/M System for the microcomputers the class is using. Show the videotape, "Hardware and Software," and provide questions for discussion, such as from Handout #2 - Questions for the Videotape, "Hardware and Software."

3. Using a demonstration microcomputer, compare the software features of a compiler versus an interpreter via a simple program coded in BASIC and the Pascal programming languages. (Be sure to use a version of Pascal, such as Apple Pascal, that does function with a compiler.) Deliberately put some syntax errors in both programs. Consider the "pros" and "cons" of both systems. Two examples of a program coded in both languages are provided in Handout #3 - Comparison of Two Computer Languages.

4. Review or introduce the concepts of bit, byte and binary system. Distinguish the major data types--ASCII characters, integers and floating point numbers. Distribute Handout #4 - Binary Card Game and have students perform the game. The cards could already be prepared in advance for the students, but the important point is for each individual to try using the cards as directed and explain why the results turn out as they do. This could lead to an interesting class discussion.

Sample Assignments and Materials for Activity Cluster #1:

The following pages in this activity cluster provide handouts that could be used in this course.

Handout #1
Review

NAME_____
DATE_____
PERIOD_____

COMPUTER LAB RULES

1. The lab will be available for academic use from 7:00 a.m. to 3:00 p.m. Plan your work accordingly.

2. When needing the lab outside of class time to work on an assignment, sign up for only one time slot at a time--either before school, at recess, lunch or after school. If you cannot make that appointed time, cross your name off the sign-up list. If you are late for that appointment beyond five minutes, you forfeit that lab time, unless no one else needs it.

3. Non-academic game-playing is not allowed.

4. No food and drinks are allowed in the lab.

5. No TV watching is permitted.

6. No copying is permitted. Anyone found copying will forfeit all future use of the computer lab outside of class time. Only one backup copy of your own work disk is permitted.

7. Do not tamper with other students' disks stored in the lab. Written permission by that student to use his or her disk is not acceptable. An appropriate consequence will follow any such violation.

8. Strive to be polite and considerate of others in the computer lab. Keep the noise level down, and vacate your position immediately upon completing your time slot.

Handout #2
Review

NAME_____
DATE_____
PERIOD_____

QUESTIONS FOR THE VIDEOTAPE
"Hardware and Software"

1. How do competitors sneak into Atari's developmental lab in the Silicon Valley?

2. Why is there a software lag? What does this mean for the future?

3. What role does a modem play in data communications?

4. Distinguish the differences among mainframes, minicomputers and microcomputers.

5. Name three devices for inputting data into a computer system and three devices for outputting data from this system.

6. In what three ways can printers be characterized or described?

7. What are the five components of any computer system? Describe the relationships among these components.

8. How are systems programs different from systems services? Where does the operating system fit?

9. What converts source code to object code or machine language?

10. Why will non-procedural languages become the ultimate in user-friendly computer languages?

Handout #3
Review

NAME_____
DATE_____
PERIOD_____

COMPARISON OF TWO COMPUTER LANGUAGES

BASIC

(Uses an Interpreter to Translate into Machine Language)

```
     RECAREA
05   REM Area of Rectangle
10   LET L = 17
20   LET W = 12
30   LET A = L*W
40   PRINT L,W,A
50   END
```

```
     COUNT5
05   REM Increments by 5
10   FOR X = -10 TO 10 STEP 5
20   PRINT X
30   NEXT X
40   END
```

or

```
05   REM Increments by 5
10   LET X = -15
20   LET X = X + 5
30   PRINT X
40   IF X < 10 THEN 20
50   END
```

PASCAL

(Uses a Compiler to Translate into Machine Language)

```
PROGRAM AREA;
VAR LENGTH, WIDTH, AREA: REAL;
BEGIN
  LENGTH:= 17.0;
  WIDTH:= 12.0;
  AREA:= LENGTH*WIDTH;
  WRITELN (LENGTH, WIDTH, AREA)
END.
```

```
PROGRAM COUNT5;
VAR X: REAL;
BEGIN
 X:= -15.0;
 REPEAT
  X:= X + 5.0;
  WRITELN (X)
 UNTIL X = 10.0
END.
```

NAME _____
DATE _____
PERIOD _____

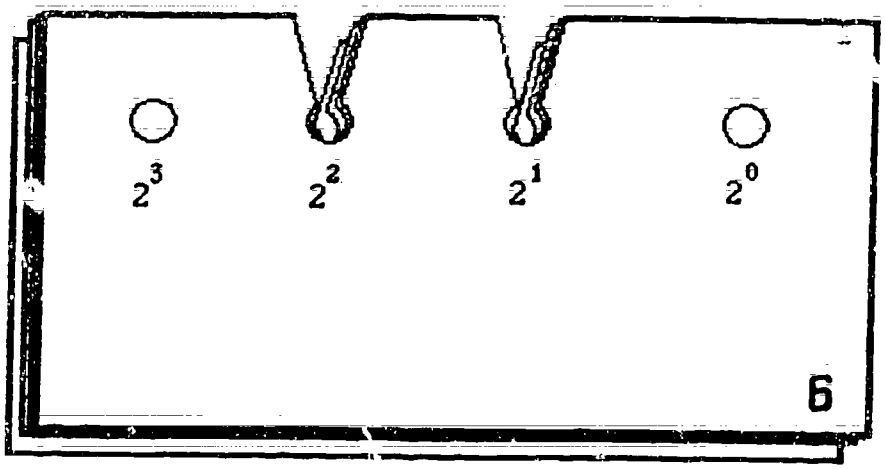
BINARY CARD GAME

ESSENTIAL ELEMENT ADDRESSED: Communicating instructions to the computer; Using the computer as a tool (sorting)

MATERIALS: 15 3"x5" index cards per student
Scissors
Hole Punch
Black marker
Extended paper clip

INSTRUCTIONS FOR MAKING BINARY CARDS:

1. Punch 4 holes 1" apart on the 5" side of the index cards.
2. Each hole will represent a place value in base two numeration.
3. Under each hole, write 2 raised to appropriate powers in order from right to left.
4. Number each card (using decimal numeration) 1-15 on the lower right corner.
5. Designate the value of the number written on the card by clipping appropriate holes. Clipped holes represent a 1, and closed holes represent a 0. In the example below, the decimal number 6 is shown, since 110(base two) = 6(base ten).



Note: Each pair of students should construct his or her own set of 15 cards.

INSTRUCTIONS FOR USE:

1. Shuffle the cards to mix up their order.
2. Stack all 15 cards, with punched edge of cards up.
3. Extend a paper clip and insert through the first hole on the right hand side. All cards that have a closed hole in the one's place will remain on the paper clip. All others will drop off.
4. Place the cards which drop off at the back of the stack.
5. Now put the paper clip through the twos place and follow the same procedure.
6. Continue this process through and including the eights place hole. The cards should be in numeric order.

SAMPLE ACTIVITY CLUSTER #2

Structured Programming via Subroutines

Topics:

1. Data Statements
2. Subroutines and Iteration Control Structures
3. Top-Down Design
4. Low-Resolution Graphics
5. Bar-Graphs

Classroom Management:

Some of the work is best done by the class as a whole; others by pairs of students; and still others by individuals.

Materials:

Demonstration microcomputer and large-screen monitor with demo programs, "Making a Face" and "Bar Graph" on disk;
Handouts;
Current Almanacs.

Time for Activities:

Approximately two weeks

Teacher Preparation and Procedures:

Prepare the handouts and sample programs on disk. All sample materials for these activities are located in the Sample Assignments and Materials for Activity Cluster #2.

1. Using the chalkboard, review the basics of READ-DATA statements with numerous examples having errors. Let the class as a whole find the syntax and logic errors.

2. Introduce or review (depending on the previous experience of the class) subroutines and low-resolution graphics via a lecture and demonstration, using the demo microcomputer and the "Making a Face" program already on disk. Distribute Handout #1 - Demo Face Subroutine Program for students to see a listing of the program on disk. Emphasize the concepts of top-down design and modular programming, and have students interpret the statements. After running the program, have groups of students modify it to generate six faces on the screen and then to display the faces in random colors.

3. Assign two or three short programs from Handout #2 - Data Statements and Subroutines. Have students work in pairs.

4. Using the microcomputer to further demonstrate subroutines, present a program for drawing a bar graph, not using low-resolution graphics but utilizing READ-DATA and GOSUB-RETURN statements. Have the class help develop the program. After distributing Handout #3 - Demo Bar Graph Program, have the class modify it by incorporating names of different lengths in the DATA statement, scale the numeric data and line up the starting asterisks.

5. Assign a bar graph program using Subroutines and Data Statements, as in the demonstrated bar graph program. Provide several levels of difficulty and extra-credit work and assign each student a program at the appropriate level. Allow students to look up their own data in the library or from classroom Almanacs. Provided in Handout #4 - Bar Graph, Using Subroutines and Data Statements are six sample bar graph assignments with varying levels of difficulty for different students. Each offers the same extra credit assignment, however.

Sample Assignments and Materials for Activity Cluster #2:

The following pages in this activity cluster provide handouts that could be used in this course.

Handout #1
Structured Programming

NAME _____
DATE _____
PERIOD _____

DEMO FACE SUBROUTINE PROGRAM*

```
0001  REM ** MAKING A FACE **
0002  REM ** GRAPHIC SUBROUTINE **
0003  REM **************
0010  HOME
0020  GR
0030  COLOR= 1
0040  REM **MAIN PROGRAM**
0050  X = 0:Y = 0:  GOSUB 1000
0060  X = 20:Y = 0:  GOSUB 1000
0070  FOR I = 1 TO 2500: NEXT I
0080  TEXT : HOME
0090  GOTO 2000
1000  REM **FACE SUBROUTINE**
1010  HLIN X,X + 9 AT Y
1020  VLIN Y + 1,Y + 8 AT X + 9
1030  HLIN X + 9,X AT Y + 9
1040  VLIN Y + 8,Y + 1 AT X
1050  PLOT X + 3,Y + 2: PLOT X + 6,Y + 2
1060  HLIN X + 4,X + 5 AT Y + 4
1070  PLOT X + 2,Y + 5:  PLOT X + 3,Y + 6
1080  HLIN X + 4,X + 5 AT Y + 7
1090  PLOT X + 6,Y + 6:  PLOT X + 7,Y + 5
1100  RETURN
2000  END
```

*From Apple BASIC by Richard Haskell, page 79; copyright (c) 1982 by Prentice-Hall, Inc., Englewood Cliffs, New Jersey. Reproduced with permission.

NAME _____
DATE _____
PERIOD _____

DATA STATEMENTS AND SUBROUTINES

1. A freight company charges $70 per ton for the first 12 tons and $40 per ton for every ton over 12. The following companies had shipments as indicated:

Company A	14 tons
Company B	42 tons
Company C	6 tons
Company D	130 tons
Company E	2360 tons

Write a program using DATA statements and SUBROUTINES to print out the company name, the tons shipped and the charges for each company.

2. Several schools ordered textbooks from the mainland. The charges for the books were: under 30 copies, $12.95 each and 30 or more copies, $12.75 each.

School A	35 copies
School B	70 copies
School C	12 copies
School D	90 copies
School E	25 copies

There is also a $0.15 per book shipping charge. Write a program using DATA statements and SUBROUTINES to print out the total bill for each school.

3. A company computes the monthly earnings of a salesman on the following basis: monthly earnings are 18% of total sales, plus a bonus of 15% of any amount sold in excess of $6000. There are 6 salesmen in the company, and for the month of January their sales were as follows:

SALESMAN	MONTHLY SALES
1	$ 8200
2	4800
3	6800
4	7850
5	5560
6	5900

Put the amount of sales in DATA statements and write a program using SUBROUTINES to calculate each person's monthly earnings, including any bonuses earned.*

*From Data Processing: An Introduction with BASIC by Donald D. Spencer, page 450; copyright (c) 1982 by Charles E. Merrill Publish g Co., Columbus, Ohio. Reproduced with permission.

Handout #3
Structured Programming

NAME _____
DATE _____
PERIOD _____

DEMO BAR GRAPH PROGRAM

```
001  HOME
003  PRINT "BAR GRAPH"
005  GOSUB  550
007  PRINT
015  FOR I = 1 TO 5
020  READ A,A$
022  PRINT A$;" ";
025  GOSUB 500
030  NEXT I
050  GOSUB 550
300  GOTO 700
499  REM ROUTINE TO PRINT *
500  FOR J = 1 TO A
510  PRINT "*";
520  NEXT J
530  PRINT:PRINT
540  RETURN
549  REM ROUTINE TO PRINT -
550  FOR I = 1 TO 40:  PRINT "-";
560  NEXT I
570  RETURN
600  DATA 29,A,13,B,2,C,17,D,6,E

700  END
```

OUTPUT:

```
BAR GRAPH
----------------------------------------
A ******************************
B *************
C **
D ******************
E ******
----------------------------------------
```

Change line 600 to:

600 DATA 60, Art, 90, Bob, 30, Charles, 50, David, 20, Edward

Modify the program to scale the numeric data and line up the starting *.

NAME _____
DATE _____
PERIOD _____

BAR GRAPH - USING SUBROUTINES AND DATA STATEMENTS

COURSE	NUMBER (have taken or are now in class)
Core Alg	
Alg I	
Geometry	
Alg II	
Trig/Analyt	
Calculus	
Computer Math	

Poll the members of the class and get the number of students who have taken each of the above courses.

Complete this chart and store the information in Data statements. Plot the results in BAR GRAPH form. Save as BARGRAPH.

EXTRA CREDIT

BAR GRAPH - USING SUBROUTINES AND DATA STATEMENTS

ISLAND	AREA(SQUARE MILES)
Kauai	
Maui	
Molokai	
Oahu	
Lanai	
Hawaii	

Complete this chart and store the information in Data statements. Plot the results in BAR GRAPH form. Save as BGRAPH2.

NAME_____
DATE_____
PERIOD_____

BAR GRAPH - USING SUBROUTINES AND DATA STATEMENTS

TEAM	GAMES WON (YEAR)
BYU	
Air Force	
Hawaii	
Colorado St	
Utah	
Wyoming	
New Mexico	
San Diego St	
UTEP	

Complete this chart and store the information in Data statements. Plot the results in BAR GRAPH form. Save as BARGRAPH.

EXTRA CREDIT

BAR GRAPH - USING SUBROUTINES AND DATA STATEMENTS

ISLAND	AREA(SQUARE MILES)
Kauai	
Maui	
Molokai	
Oahu	
Lanai	
Hawaii	

Complete this chart and store the information in Data statements. Plot the results in BAR GRAPH form. Save as BGRAPH2.

NAME _____
DATE _____
PERIOD _____

BAR GRAPH - USING SUBROUTINES AND DATA STATEMENTS

STATE	POPULATION
New York	
New Jersey	
California	
Pennsylvania	
Massachusetts	
Texas	
Illinois	

Complete this chart and store the information in Data statements. Plot the results in BAR GRAPH form. Save as BARGRAPH.

EXTRA CREDIT

BAR GRAPH - USING SUBROUTINES AND DATA STATEMENTS

ISLAND	AREA(SQUARE MILES)
Kauai	
Maui	
Molokai	
Oahu	
Lanai	
Hawaii	

Complete this chart and store the information in Data statements. Plot the results in BAR GRAPH form. Save as BGRAPH2.

NAME _____
DATE _____
PERIOD _____

BAR GRAPH - USING SUBROUTINES AND DATA STATEMENTS

STATE	AREA (SQUARE MILES)
Washington	
Oregon	
California	
Arizona	
Nevada	
Idaho	
Utah	
.	
Alaska	

Complete this chart and store the information in Data statements. Plot the results in BAR GRAPH form. Save as BARGRAPH.

EXTRA CREDIT

BAR GRAPH - USING SUBROUTINES AND DATA STATEMENTS

ISLAND	AREA(SQUARE MILES)
Kauai	
Maui	
Molokai	
Oahu	
Lanai	
Hawaii	

Complete this chart and store the information in Data statements. Plot the results in BAR GRAPH form. Save as BGRAPH2.

NAME _____
DATE _____
PERIOD _____

BAR GRAPH - USING SUBROUTINES AND DATA STATEMENTS

PLANET	MILES FROM SUN
Earth	
Mercury	
Venus	
Mars	
Jupiter	
Uranus	
Saturn	
Neptune	
Pluto	

Complete this chart and store the information in Data statements. Plot the results in BAR GRAPH form. Save as BARGPAPH.

EXTRA CREDIT

BAR GRAPH - USING SUBROUTINES AND DATA STATEMENTS

ISLAND	AREA(SQUARE MILES)
Kauai	
Maui	
Molokai	
Oahu	
Lanai	
Hawaii	

Complete this chart and store the information in Data statements. Plot the results in BAR GRAPH form. Save as BGRAPH2.

NAME _____
DATE _____
PERIOD _____

BAR GRAPH - USING SUBROUTINES AND DATA STATEMENTS

NATION	POPULATION	AREA(SQUARE MILES)
Australia		
Bangladesh		
Britain		
Canada		
China		
India		
Japan		
USA		
USSR		

Complete this chart and store the information in Data statements. Have the computer calculate the population density for each country. Plot the results in BAR GRAPH form. Save as BARGRAPH. (NOTE: density = population/area)

EXTRA CREDIT

BAR GRAPH - USING SUBROUTINES AND DATA STATEMENTS

ISLAND	AREA(SQUARE MILES)
Kauai	
Maui	
Molokai	
Oahu	
Lanai	
Hawaii	

Complete this chart and store the information in Data statements. Plot the results in BAR GRAPH form. Save as BGRAPH2.

SAMPLE ACTIVITY CLUSTER #3

Subscripted Variables and Problem-Solving Strategies

Topics:

1. One-dimensional Arrays
2. Two-dimensional Arrays
3. Problem Definition and Flowchart Design
4. Control Structures and Logical Operators
5. Error-Checking Subroutines
6. Documentation
7. Testing and Debugging Techniques
8. Matrix Manipulation (Optional)
9. Hierarchy Charts and Menus

Classroom Management:

Some of the work is best done by the class as a whole, by pairs of students and by individuals.

Materials:

Videotape player and the videotape, "Subscripted Variables and Arrays";
Demonstration microcomputer and large-screen monitor with the demo program, "Visible Bubble Sort" from Softdisk Magazine and one menu-driven program, such as "Summer Games" from EPYX Computer Software;
Overhead projector and transparency for hierarchy chart;
Handouts.

Time for Activities:

Approximately four to five weeks

Teacher Preparation and Procedures:

Prepare the handouts and transparency and preview the demo program. All sample materials for these activities are located in the Sample Assignments and Materials for Activity Cluster #3.

1. Using the chalkboard or overhead projector, introduce or review one-dimensional arrays. Show how they are more effective than READ-DATA Statements in manipulating large amounts of data. Assign any related reading material.

2. Remind students about the careful use of the logical operators--AND and OR in the IF-THEN control structure. Provide Handout #1 - IF-THEN Control Structure as a worksheet on predicting output using the IF-THEN Control Structure and logical operators in problem solving. Review the differences between the IF-THEN and the FOR-NEXT control structures. Have students incorporate these control structures in a program which performs the divisibility tests for 2, 3, 4, 5, 6, 8, 9 and 10 for some input number. A sample listing

for one solution to this problem, using Applesoft BASIC, is provided in Handout #2 - Divisibility Program.

3. Distribute to pairs of students Handout #3 - Array Assignment 1 with several short array problems. Emphasize the need for continuing the problem-solving strategies learned in BASIC Programming I, such as problem definition and flowchart design. Continue to stress the importance of structured programming, using the proper control structures of sequence, selection and/or iteration.

4. For review and further explanation, show the videotape, "Subscripted Variables and Arrays," and provide Handout #4 - Questions for the Videotape, "Subscripted Variables and Arrays," for discussion. This film presents a good introduction to the Bubble Sort.

5. Have students individually work on Handout #5 - Subscript Worksheet to determine the output for several subscript problems. Discuss results. This worksheet may be most appropriate as a "break" in the Array Assignment 1 that reinforces student understanding of subscripted variables.

6. Present the bubble sort via lecture and the demo computer program, "Visible Bubble Sort" (from Softdisk Magazine). For further information on this program, see page 198. Have students select one of three programs to develop from Handout #6 - Bubble Sort Assignment.

7. Provide a review worksheet, Handout #7 - GOSUB, for determining the output of various subroutines. Assign Handout #8 - Crunch for fun and also as a review for designing a structured program, using subroutines.

8. Introduce two-dimensional arrays via a lecture and group demonstration of loading and displaying a table having four rows and five columns. Distribute Handout #9 - Array Assignment 2 with several programming problems from which one is chosen by each student. Flowcharts should be required as well as good documentation by including appropriate REM statements. If necessary, provide Handout #10 - Review Worksheet on Arrays. Handout #11 - Test on Arrays may also be helpful.

9. Demonstrate the value of providing error-checking or error-trapping subroutines for valid input data. An example of such subroutines is provided in Handout #12 - Error-Checking Subroutine. Stress that all subsequent programs should have entry validation subroutines.

10. Review the procedures for testing and debugging structured programs. Excellent examples of programs needing debugging are found in Handout #13 - Debugging Exercise. (Refer to the restriction on reproduction, regarding these exercises.) Consider the tracing techniques provided by the computer system in use. Show how the PRINT statement can be temporarily inserted to test

variables, especially after calculations, during search subroutines or before selection control structures.

11. As an optional activity, if your computer system accepts MAT statements, demonstrate the use of MAT-READ, MAT-PRINT and MAT-INPUT statements as matrix operations, which can simplify the handling of one- and two-dimensional arrays. Two example programs are provided in Handout #14 - MAT Statements. The text, The Basics of BASIC by Alfredo C. Gomez, offers an excellent chapter on Matrices. Challenge the more adept students with matrix addition, subtraction or multiplication problems.

12. Using Handout #15 - Hierarchy Chart, present a lecture with the aid of an overhead projector to show how hierarchy charts:

    a. encompass the basic computer processing steps of input--processing--output;
    b. aid in the visualization of top-down structured programming;
    c. break a program into small, easy-to-write and debug modules.

    From the same handout assign programs for designing hierarchy charts and have the class identify and develop hierarchy charts for each program. Assign programs to be flowcharted and solved, using the modules identified as subroutines. Explain the "stubbing-in technique," as described on pages 20-21 of Advanced Structured BASIC by Clark and LaBarre.

13. Demonstrate software that is menu-driven. (Any available software can be used since almost all user programs are menu-driven, such as word processing, problem-solving games etc.) Have the class discuss the advantages of menu-driven programs for the user and programmer. Present a lecture on the relationship between a hierarchy chart and a menu and on how menus are developed and used in programming. Explain the use of ON GOTO and ON GOSUB, verification of selection from menu, and termination of program.

14. Distribute Handout #16 - Menu-Driven Programming Assignments. Discuss with the class possible menu items for each program. Assign programs to be written by individuals or pairs. A sample solution to problem 1 for the TRS-80 Model 4 and IBM PC computers is included in the handout.

Sample Assignments and Materials for Activity Cluster #3:

The following pages in this activity cluster provide handouts that could be used in this course.

NAME _____
DATE _____
PERIOD _____

IF-THEN CONTROL STRUCTURE

An IF...THEN statement will cause control of the program to branch to the designated line when the given expression is TRUE. Sometimes it may be difficult determining whether a branch will occur. Here is a short program segment that may help you decide.

```
40  X=12:Y=2
50  IF X*2 > 5 THEN 80
60  PRINT "IT DID NOT BRANCH"
70  GOTO 90
80  PRINT "IT DID BRANCH"
90  END
```

Listed below are a number of IF...THEN statements that should replace line 50 in the program above. Your job is to decide whether a branch to line 80 WILL or WILL NOT take place when X=12 and Y=2. Circle the correct answer. If you think the computer will produce a SYNTAX ERROR then circle the entire problem.

A)	50 IF X not = Y THEN 80	will	will not
B)	50 IF X+Y/X > 10 THEN 80	will	will not
C)	50 IF X-Y > 10 or X < 10 THEN 80	will	will not
D)	50 IF X/Y - INT(X/Y) > .0001 THEN 80	will	will not
E)	50 IF X/(Y-INT(X/Y)) > .0001 THEN 80	will	will not
F)	50 IF INT(-1*X+3.7)<>-44 THEN 80	will	will not
G)	50 IF INT((Y+.5)*10+.5)/10=2.5 THEN 80	will	will not
H)	50 IF X<100 OR Y<10 AND X>15 THEN 80	will	will not
I)	50 IF (X<100 or Y<10) AND X>15 THEN 80	will	will not
J)	50 IF X/INT(-.4) > Y THEN 80	will	will not
K)	50 IF X<>Y THEN 80	will	will not

L)	50 IF 12 >=X THEN 80	will	will not
M)	50 IF X/Y <> INT(X/Y) THEN 80	will	will not
N)	50 IF X<10 THEN PRINT "GEE!":GOTO 80	will	will not

NAME_____
DATE_____
PERIOD_____

```
010  HOME
020  REM        DIVISIBILITY PROGRAM
025  REM        SAMPLE SOLUTION
030  REM ************************************
040  REM *      ASSIGNMENT OF VARIABLES       *
045  REM ************************************
050  REM        K$ = RESPONSE VARIABLE TO BEGIN PROGRAM
060  REM        N$ = INPUT NUMERAL
070  REM        N  = VALUE OF INPUT NUMBER
080  REM        T1$= ONES DIGIT OF INPUT NUMERAL
090  REM        T1 = VALUE OF THE ONES DIGIT
100  REM        T2$= TENS DIGIT OF INPUT NUMERAL
110  REM        T2 = VALUE OF THE TENS DIGIT
120  REM        T3$= HUNDREDS DIGIT OF INPUT NUMERAL
130  REM        T3 = VALUE OF THE HUNDREDS DIGIT
140  REM        R$ = DIFFERENCE OF LOOP NUMERAL
150  REM        RN = VALUE OF THE RESULTING DIFFERENCE IN LOOP
160  REM        P  = NUMBER OF DIGITS IN INPUT NUMERAL
170  REM        M  = LOOP VARIABLE
180  REM        D1$= LEFT MOST DIGIT OF THE DIFFERENCE IN LOOP
190  REM        D1 = VALUE OF THE LEFT MOST DIGIT IN LOOP
200  REM        PV = PLACE VALUE OF LEFT MOST DIGIT IN LOOP
210  REM        S  = SUM OF THE DIGITS OF THE NUMERAL
220  REM        A$ = RESPONSE TO TRY AGAIN
225  REM ***************************
230  REM *      TITLE SCREEN         *
235  REM ***************************
240  VTAB (7): HTAB (4)
250  INVERSE
260  PRINT "  TEST OF DIVISIBILITY FOR "
270  NORMAL
280  PRINT : PRINT
290  PRINT "   2, 3, 4, 5, 6, 8, 9, and 10"
300  VTAB (18)
310  PRINT "   PRESS ANY KEY WHEN YOU ARE READY ": GET K$
314  REM ************************************
315  REM *      INPUT NUMBER TO BE TESTED     *
316  REM ************************************
320  HOME
330  PRINT "  ENTER A NUMBER ";
340  INPUT N$
345  REM ****************************************
350  REM *      INITIALIZE THE SUM VARIABLE      *
355  REM ****************************************
360  LET S = 0
```

```
364  REM ***********************************************
365  REM *      CONVERTING STRING TO NUMERIC VALUE     *
366  REM *      AND FINDING KEY DIGITS                 *
367  REM ***********************************************
370  LET N = VAL (N$)
380  REM
390  LET T1$ = RIGHT$ (N$,1)
400  LET T1 = VAL (T1$)
410  REM
420  LET T2$ = RIGHT$ (N$,2)
430  LET T2 = VAL (T2$)
440  REM
450  LET T3$ = RIGHT$ (N$,3)
460  LET T3 = VAL (T3$)
465  REM
467  REM ****************************************
468  REM *      LOOP FOR SUMMATION OF DIGITS      *
469  REM ****************************************
470  LET R$ = N$
475  LET P = LEN (R$)
480  FOR M = P TO 1 STEP - 1
490  LET P = LEN (R$)
500  LET D1$ = LEFT$ (R$,1)
510  LET D1 = VAL (D1$)
520  LET PV = D1 * 10 ^ (M - 1)
530  LET N = N - PV
535  IF INT (N) - N<0 THEN N = INT (N) + 1
540  LET R$ = STR$ (N)
550  LET S = S + D1
560  NEXT M
570  PRINT : PRINT : PRINT
575  REM
576  REM *********************************
577  REM *      DIVISIBILITY TESTS       *
578  REM *********************************
580  IF T1 / 2 = INT (T1 / 2) THEN PRINT N$;" IS DIVISIBLE BY 2."
590  IF S / 3 = INT (S / 3) THEN PRINT N$;" IS DIVISIBLE BY 3."
600  IF T2 / 4 = INT (T2 / 4) THEN PRINT N$;" IS DIVISIBLE BY 4."
610  IF T1 = 0 OR T1 = 5 THEN PRINT N$;" IS DIVISIBLE BY 5."
620  IF S / 3 = INT (S / 3) AND T1 / 2 = INT (T1 / 2) THEN PRINT N$;" IS
     DIVISIBLE BY 6."
630  IF T3 / 8 = INT (T3 / 8) THEN PRINT N$;" IS DIVISIBLE BY 8."
640  IF S / 9 = INT (S / 9) THEN PRINT N$;" IS DIVISIBLE BY 9."
650  IF T1 = 0 THEN PRINT N$;" IS DIVISIBLE BY 10."
659  REM  ENCOUNTERED PRINTOUT PROBLEMS FOR 660 SO 661 AND 662 WERE ADDED
660  IF T1 / 2 <> INT (T1 / 2) AND S / 3 <> INT (S / 3) AND T1 <> 0 AND
     T1 <> 5 THEN
661  GOTO 665
662  PRINT N$;" IS NOT DIVISIBLE BY ANY OF THE NUMBERS TESTED."
```

```
665  REM ********************************
670  REM *      OPTION TO CONTINUE      *
675  REM ********************************
680  PRINT : PRINT : PRINT : PRINT
690  PRINT " WOULD YOU LIKE TO TRY ANOTHER NUMBER";
700  INPUT A$
710  IF A$ = "YES" OR A$ = "Y" THEN GOTO 320
720  IF A$ = "NO" OR A$ = "N" THEN GOTO 790
730  PRINT : PRINT
740  PRINT "PLEASE RESPOND WITH EITHER"
750  PRINT "       'YES' OR 'NO'"
760  FOR T = 1 TO 1000: NEXT T
770  HOME
780  GOTO 690
785  REM ************************
786  REM *      ENDING SCREEN     *
787  REM ************************
790  HOME
800  VTAB (7): HTAB (10)
810  PRINT "HAVE A NICE DAY."
900  END
```

NAME _____
DATE _____
PERIOD _____

ARRAY ASSIGNMENT 1

Prepare a flowchart for each of the following problems before entering code into the computer.

1. Load 10 values into an array A and 10 values into an array B, and construct and output an array C for which every element in C is the sum of the corresponding elements in arrays A and B. (Save as ARRAY1.)*

2. Often a programmer is required to search through the elements in an array to find a specific or "target" value, the number of times it appears or the location of the target value. For example, load a 10-element array M and search the array to find the number of zero elements in it. (Save as ARRAY2.)*

3. Modify problem #2 and search the array M for the positions of the zero elements in the array. This problem seeks the subscript designating the location in M of each zero value. (Save as ARRAY3.)*

4. Search an array R of 12 values for the largest value stored in it. Output this value and its location. (Save as ARRAY4.)*

5. Write a program to read the numbers 15, 63, 42, 87, 65, 99, 18 into array X, and the numbers 84, 63, 44, 19, 98, 15, 87 into array Y. The program should form and print a new list that contains only those numbers that are in both lists. (Save as ARRAY5.)**

*From Programming Apple BASIC by John J. Dielsi, Elaine S. Grossman, John P. Tucciarone, pages 227-230; copyright 1984 by CBS College Publishing Co., New York, New York. Reproduced with permission.

**From Data Processing: An Introduction with BASIC by Donald D. Spencer, page 481; copyright (c) 1982 by Charles E. Merrill Publishing Co., Columbus, Ohio. Reproduced with permission.

6. The factorial of a number is calculated by multiplying the number by successive smaller integers until the number 1 is reached. Here are some examples:

5!=5x4x3x2x1 or 120
7!=7x6x5x4x3x2x1 or 5040
3!=3x2x1 or 6

Write a program to accept numbers from the keyboard, store them into an array, send them to a subroutine to calculate the factorial and print out a table similar to the one below. The number 999 will be the signal to halt inputs. (Save as ARRAY6.)

An example output would look as follows:

5 factorial = 120
7 factorial = 5040
3 factorial = 6

END OF LISTING

Handout #4
Variables and Strategies

NAME _____
DATE _____
PERIOD _____

QUESTIONS FOR THE VIDEOTAPE,
"Subscripted Variables and Arrays"

1. What is meant by a string variable?

2. How does L$ = "3x5" differ from L = 3x5?

3. What is meant by subscripted variables?

4. Distinguish the difference among one-dimensional, two-dimensional and three-dimensional arrays.

5. Illustrate how a FOR-NEXT loop can load an array.

6. What is a Bubble Sort?

7. Why is a temporary storage area needed in a Bubble Sort?

8. Why is it a good idea to use a subroutine for sorting?

Handout #5
Variables and Strategies

NAME _____
DATE _____
PERIOD _____

SUBSCRIPT WORKSHEET*

Determine the output:

1.
```
10  DIM X(12)
20  FOR L = 1 TO 12
30  X(L) = L + 1
40  NEXT L
50  FOR J = 1 TO 2
60  PRINT X(J) + X(J + 1) + X(J + 2),
70  NEXT J
80  END
```

2.
```
10  DIM A(12)
20  FOR K = 1 TO 12
30  A(K) = 2 * K
40  NEXT K
50  FOR K = 1 TO 3
60  PRINT A(K) + A(K + 2) + A(K + 4)
70  NEXT K
80  END
```

3.
```
10  DIM N(12)
15  S = 0
20  FOR K = 1 TO 12
30  N(K) = K + 4
40  NEXT K
50  FOR L = 12 TO 10 STEP -1
60  S = S + N(L)
70  PRINT S,
80  NEXT L
90  END
```

4.
```
10  DIM A(10), B(10)
20  FOR J = 1 TO 10
30  A(J) = J + 3
40  B(J) = J + 7
50  NEXT J
60  FOR K = 1 TO 10 STEP 2
70  PRINT A(K) + B(11-K)
80  NEXT K
90  END
```

5.
```
10  DIM X(10)
20  FOR N = 3 TO 6
30  X(N - 2) = N + 5
40  X(N + 3) = 2 * N
50  NEXT N
60  FOR N = 1 TO 5 STEP 2
70  PRINT X(N) + X(N + 3) +
    X(N + 5),
80  NEXT N
90  END
```

6.
```
10  DIM B(5)
15  FOR P = 1 TO 5
20  B(P) = 5 * P
25  NEXT P
30  FOR A = 1 TO 5
35  B(A) = B(6 - A)
40  NEXT A
45  FOR F = 1 TO 3
50  PRINT B(F) - F,
55  NEXT F
60  END
```

Handout #6
Variables and Strategies

NAME_____
DATE_____
PERIOD_____

BUBBLE SORT ASSIGNMENT

Do one of the following three programs. The assignment is worth 50 points, and the maximum number of points you can earn for each of the programs is listed before the problem. Save as BUBBLEA, BUBBLEB or BUBBLEC.

38 pts A) Using the following DATA statement, write a program to read in and sort the data alphabetically. The program should print out the original list and the sorted list. A subroutine should be used for the sorting.

Use this data statement.

```
1000 DATA TOM, DICK, HARRY, BOB, MARY, ANN, SUE, ANDY, BOBBIE,
     MARIE
```

42 pts B) Write a program that will accept a list of names typed in from the keyboard. The program should print out this list and then print it out in alphabetical order. The user should first be asked how many names he or she will input. A subroutine must be used for the sorting.

50 pts C) Write a program that will accept a person's name and his or her test score into two arrays P$(X) and S(X) from the keyboard. The program should allow the data to be sorted in either alphabetical order or in test score order. The user must be allowed to choose the sort order. The output should be similar to the samples listed below. Use subroutines for the sorts.

ALPHABETICAL		TEST SCORE
ANN 75		BOB 82
BOB 82	or	ANN 75
SUE 70		SUE 70

NAME_____
DATE_____
PERIOD_____

GOSUB*

Determine the output

1.
```
10 X = 5
15 GOSUB 35
20 GOSUB 45
25 GOSUB 40
30     GOTO 60
35 X = X + 4
40 X = X + 9
45 X = X + 7
50 PRINT X,
55 RETURN
60 END
```

_____

2.
```
10 GOSUB 50
15 GOSUB 40
20 GOSUB 35
25 GOSUB 60
30     GOTO 65
35 PRINT "TI",
40 PRINT "ME",
45 RETURN
50 PRINT "SO",
55 RETURN
60 PRINT "S"
65 END
```

_____

3.
```
10 GOSUB 40
15 GOSUB 60
20 PRINT X,
25 GOSUB 40
30 PRINT X,
35     GOTO 70
40 X = X + 5
45 GOSUB 60
50 PRINT X,
55 RETURN
60 X = X + 7
65 RETURN
70 END
```

_____

4.
```
10 A = 11
15 GOSUB 40
20 PRINT A,
25 GOSUB 45
30 PRINT A
35     GOTO 70
40 A = A - 3
45 A = 2 * A
50 GOSUB 60
55 A = 2 * A
60 A = A - 4
65 RETURN
70 END
```

_____

5.
```
10 FOR B = 1 TO 4
20 READ D$, D
30 GOSUB 60
40 NEXT B
50       GOTO 99
60 IF D/2 = INT(D/2) GOTO 90
70 PRINT D$;
80 PRINT "OOL";
90 RETURN
95 DATA T, 18, O, 7, F, 8.2, P, 2
99 END
```

*From Duplicating Masters - Experiencing BASIC by Michael Mulcahy, page 10; copyright (c) 1984 by Media Materials, Inc., Baltimore, Maryland. Reproduced with permission.

NAME_____
DATE_____
PERIOD_____

CRUNCH*

In the game of Crunch, 21 sticks prop open the mouth of an imaginary alligator. You and the computer take turns removing sticks from the alligator's mouth. You may take 1, 2, 3 or 4 sticks each turn. The one who removes the last stick causes the mouth of the alligator to snap shut and loses.

CRUNCH RULES:

Start with 21 sticks

You take 1, 2, 3, or 4 sticks

Computer check to see if your move is legal - in this game it means 1, 2, 3, or 4 sticks

Subtract sticks you took

Calculate computer's move (5 minus your move)

Print out computer's move

Subtract computer's move from total remaining

Check to see if there is one stick left

You have lost if there is only one stick left

Otherwise take another turn

End

Save as CRUNCH. (100 pts)

Develop a well-structured and well-documented program. You may add graphics and sound to the program.

Extra-Credit Variations:

1. Change the number of sticks in the alligator's mou
2. Change the number of sticks that may be removed at [illegible]rn;
   If either of the above changes is incorporated into [illegible] program, it will be necessary to develop a different strategy.

STRATEGY:

This is a "RIGGED" game because the computer will always win if you go first.

1. Start with 21 sticks.
2. You may take 1, 2, 3, or 4 sticks.
3. Check to see if you made a "legal" move.
4. The computer calculates its move which is 5 minus the number of sticks you took.
5. Play continues, alternating between you and the computer, until one stick is left.
6. It will be your turn to take the lone stick. C R U N C H !

The game is played in multiples of 5.

YOU TAKE	COMPUTER TAKES
1	4
2	3
3	2
4	1

---

sticks at the beginning of the round	your move plus computer's move	sticks left at the end of the round
21	5	16
16	5	11
11	5	6
6	5	1

After four rounds, 1 stick is left and it is your turn to move. YOU LOSE!!

*Adaptation of "Crunch" by Flora Reess; published in Calculators/Computers Magazine, 1978.

NAME_____
DATE_____
PERIOD_____

ARRAY ASSIGNMENT 2 (50 pts)

Do one of the following two-dimensional array problems. The maximum number of points for each example is given in parentheses. Be sure to include the appropriate REM statements for name, date, variable identification and routine documentation. Include a flowchart with your program.

(60 pts) 1. Assume that an interstate highway system connects cities A, B, C, D, E, F, G and H in that order. The distances are: A to B: 32 miles; B to C: 49 miles; C to D: 10 miles; D to E: 75 miles; E to F: 50 miles; F to G: 63 miles; and G to H: 43 miles. Write a program using a two-dimensional array to produce a mileage table, listing the total distance along the interstate between any two cities. Use one DATA statement for the seven mileages given. (Save as MILEAGE.)*

(55 pts) 2. Use the computer to play a modified game of OTHELLO. Have it randomly fill an 8 by 8 string variable array with X's and O's and print the result on the screen. The X's are for player 1 and the O's for player 2. Have the program ask the players alternately for the row and column of the opponents piece that should be flipped (changed from X to O or vice versa). All of the opponent's pieces along the horizontal or vertical line passing through the flipped piece are also flipped. For example, if player 1 flipped the O at 1,8, the board A would be changed to board B. (Save as OTHELLO.)**

(A)

	1	2	3	4	5	6	7	8
1	X	X	O	O	X	O	X	O
2	O	O	O	X	X	O	X	X
3	X	O	X	X	O	X	O	O
4	O	O	X	O	X	O	X	O
5	O	O	X	X	X	O	X	O
6	O	O	X	O	O	X	X	O
7	O	O	X	X	O	X	O	X
8	X	X	O	X	X	O	X	X

(B)

	1	2	3	4	5	6	7	8
1	X	X	X	X	X	X	X	X
2	O	O	O	X	X	O	X	X
3	X	O	X	X	O	X	O	X
4	O	O	X	O	X	O	X	X
5	O	O	X	X	X	O	X	X
6	O	O	X	O	O	X	X	X
7	O	O	X	X	O	X	O	X
8	X	X	O	X	X	O	X	X

*From Programming in Apple BASIC by John J. Dielsi, Elaine S. Grossman, John P. Tucciarone, page 258; copyright (c) 1984 by CBS College Publishing, New York, New York. Reproduced with permission.

**From A Guide to Programming in Applesoft by Bruce Presley, page 4.20; copyright (c) 1984 by Lawrenceville Press, Lawrenceville, New Jersey. Reproduced with permission.

(50 pts) 3. Write a computer program to produce the chart similar to the one below. The chart should contain the following: year, make, selling price, and total paid for the car. The (simple) interest rate and the years to pay are entered through the keyboard and may vary each time the program is run. (Save as CARSALES.)

Key board input: Years to pay
Interest rate (simple)

Input from data statements:

1980	Dodge	$ 1500
1983	Chevy	$ 2730
1967	VW	$ 1995
1932	Model T	$15821

Output:

Interest rate...12%
Years to pay....5

year	car	price	total
1980	Dodge	$ 1500	$ 2400
1983	Chevy	$ 2730	$ 4368
1967	VW	$ 1995	$ 3192
1932	Model T	$15821	$21314

(45 pts) 4. Write a program that randomly displays a seating chart for a classroom of 20 seats with four rows and five seats in each row. There are 18 students in the class. The class list should be read in from a DATA statement. Use names of 6 letters or less. Unassigned seats should say EMPTY. (Save as SEATING.)

(40 pts) 5. An inventory table contains 10 rows and 5 columns. Load the following data:

item#	#sold	cost	sale price	total profit
327	0	3.75	5.49	0
159	0	4.29	7.39	0
237	0	7.89	9.00	0
148	0	5.69	8.50	0
265	0	3.29	4.95	0
187	0	4.99	7.89	0
211	0	3.57	5.50	0
304	0	6.87	8.25	0
517	0	5.29	7.25	0
419	0	3.85	5.29	0

Update the table by entering the number sold for each item. Compute (do not enter) the total for the profit column. Profit = (#sold*(sale price - cost)). Output the entire array with suitable headings. (Save as PROFIT.)*

(40 pts) 6. The sales tax in Hawaii is 4%. Write a program to output a table of prices from 25 cents to $5.00 in steps of 25 cents, with the corresponding amounts for sales taxes. (Save as STATETAX.)

(37 pts) 7. Write a program to accept as input a 5 by 5 table G. Use whatever numbers you want in DATA statements. Display the table. Output the smallest element in G and the subscripts defining its location in the array. (Save as SMALLEST.)

(37 pts) 8. Statistics for a nine member baseball team are arranged as follows. Load the table and compute the average to 3 digits by dividing the number of hits by the number of at bats. Output the entire table. (Save as BASEBALL.)*

player#	at bats	#hits	#homers	average
8	237	73	11	0
44	354	119	27	0
12	316	89	14	0
9	289	289	14	0
22	320	107	9	0
32	288	91	11	0
17	346	101	13	0
28	276	86	4	0
11	342	108	15	0

(33 pts) 9. Construct two 3 by 4 arrays, A and B, all of whose elements are random integers between 1 and 10. Construct an array C, all of whose elements are the sums of the corresponding elements from A and B. (Save ARRAYADD.)*

(33 pts) 10. Input a 4 by 4 array A from a DATA statement, and compute and print the array B, in which each element is 5 times the corresponding element in A. (Save as ARRAYMUL.)*

---

*From Programming in Apple BASIC by John J. Dielsi, Elaine S. Grossman, John P. Tucciarone, page 256-257; copyright (c) 1984 by CBS College Publishing, New York, New York. Reproduced with permission.

NAME_____
DATE_____
PERIOD_____

REVIEW WORKSHEET ON ARRAYS (I)*

The DIMension statement reserves space for data named by subscripted variables. DIM S(12) - reserves 12 locations in the memory for the elements in array S.

Elements in an array are named by subscripted variables. The subscript distinguishes one element from another by its location in the array.

S = 12 -8 0 1 6 3 -1 -9 10 7 -14 -5

1. It is necessary to reserve space when an array contains more than _____ elements.

2. What values in array S are named by the following subscripted variables.

   S_7___ S_3___ S_{11}___ S_2___ S_6___

3. Write a subscripted variable for each of the following elements in S.

   0 ___ -1___ 6 ___ -14___ 12___

4. In BASIC the subscripts are enclosed by _____.

5. Write each of the subscripted variables in BASIC.

   V_5___ R_{12}___ T_J ___ W_{16}___ Z_{R+S}___ A_{J^2}___

   G_{K+1}___ X_{IJ}___ L_{I-5}___

6. A subscript may be a _____, a _____, or an _____.

7. A = 15 23 -15 17 -10 62 18 25 50 100 -1 13

   Write an appropriate DIM statement for the above array.

   _____

8. If I = 2, J = 3, and K = 4, find the value which each BASIC expression represents.

   A(I) + A(J) _____ A(I + I * J) + A(I * K + I) _____

   A(K) * A(I*K) _____ A(K - J) - A(J) _____

   A(J*K/I) + A(I + 6)_____ A(I + J + K) / A(2 * K) _____

9. After space is reserved for an array, the next step is storing the elements of the array in the reserved memory locations. By placing the elements of the array in a DATA statement and by using FOR - NEXT statements to generate the subscripts, a programmer transfers an array into the computer's memory.

   Insert missing statements or complete the incomplete statements in each program which will store and print arrays.

```
10  DIM                          10                           10  DIM R(12)
20  FOR I = 1 TO 15              20  FOR J = 1 TO 30        20  FOR K = 1 TO
30  READ S(I)                    30  READ V(J)               30  READ
40  PRINT S(I)                   40  PRINT V                 40  PRINT R(K)
50  NEXT I                       50  NEXT J                  50  NEXT K
60  DATA 35,.....                60  DATA 25,.....          60  DATA 15,.....
70  END                          70  END                     70  END
```

All of the rules for variables apply to subscripted variables. Arithmetic operations may be performed using subscripted variables. Strings may be named by subscripted variables.

Each program below involves using subscripted variables for accomplishing the task described in the REM statements. Most errors involve the subscripted variables. Find and correct all errors. Rewrite each program correctly.

CORRECTED PROGRAMS

10.
```
10  REM NUMBERING A
20  REM CLASS LIST
30  DIM N$(12)
40  FOR J = 1 to 12
50  READ N$(J)
60  PRINT J, N$
70  NEXT J
80  DATA SMITH,MILLER....
90  END
```

11.
```
10  REM SUMMING GRADES
20  DIM G
30  FOR G = 1 TO 20
40  READ G(I)
50  LET S = S + G(I)
60  NEXT I
70  PRINT S "IS THE SUM."
80  DATA 95,62,73,....
90  END
```

12.
```
010  REM MULTIPLY TEST SCORES AVG BY 3
015  REM MULTIPLY QUIZ SCORES AVG BY 2
020  REM EVERY THIRD SCORE - TEST SCORE
```

```
030  DIM S(12)
040  FOR J = 1 TO 12
050  IF J=3 OR J=6 OR J=9 OR J=12 THEN 80
060  LET Q = Q + S(J)
070  GOTO 90
080  LET T = T + S(J)
090  NEXT J
100  PRINT Q/8 * 2 " IS THE WEIGHTED QUIZ AVG."
110  PRINT T/4 * 4 " IS THE WEIGHTED TEST AVG."
120  DATA 85,52,91,70,.......
130  END
```

LET statements and INPUT statements may also be used to store arrays. Find the errors and rewrite the following programs correctly.

13.
```
10  REM WORD ARRAY
20  FOR I = 1 TO 25
30  PRINT "TYPE THE WORD FOR POSITION ";I
40  INPUT W(I)
50  NEXT I
60  FOR W = 1 TO 25
70  PRINT W(L);
80  NEXT I
90  END
```

14.
```
10  REM ASSIGNING POSITIONS
20  LET S(1) = 35
30  LET S(2) = 95
40  LET S3 = S(1) + S(2)
50  LET S(4) = S(1) / 5
60  FOR I = 1 TO 4
70  PRINT SI
80  NEXT I
90  END
```

15.
```
10  REM TWO ARRAYS
20  DIM S(15), V(15)
30  FOR I = 1 TO 15
40  READ S(I)
50  NEXT I
60  FOR I = 1 TO 15
70  LET V = V + S(I) * 5
80  PRINT V(I)
90  NEXT V
95  DATA 8,3,9,1,....
99  END
```

---

*From Everybody's BASIC by Kay Richardson, pages 61-62, 64; copyright (c) 1983 by Meka Publishing Co., Indianapolis, Indiana. Reproduced with permission.

NAME _____
DATE _____
PERIOD _____

REVIEW WORKSHEET ON ARRAYS (II)*

Two dimensional arrays are practical ways of storing data. A teacher may store an individual student's grades in a single dimensional array, but it would be more practical to store all of the students' scores in a two dimensional array or matrix.

	Whole Numbers											
	QUIZZES									TESTS		
Name of Student	1	2	3	4	5	6	7	8	9	10	11	
1												
2												
3												
4												
5												
6												
7												

Elements stored in a matrix are identified by the number of their rows and columns.

A student's name would be located in column one and his or her scores in columns two through twelve. The row number would identify the student.

It is easy to see that every item in a matrix needs two subscripts to identify its location.

1. Find the values of the following subscripted variables in matrix B.

$$B = \begin{vmatrix} 5 & 4 & 3 & 6 & 7 \\ 9 & 7 & 1 & 2 & 3 \\ 6 & 0 & 5 & 7 & 8 \end{vmatrix}$$

B(3,5) = ____ B(1,1) = ____ B(2,3) = ____ B(3,2) = ____

B(1,4) = ____ B(2,5) = ____ B(2,2) = ____ B(3,3) = ____

2. Insert subscripts to identify the items in the following matrix.

$$T = \begin{vmatrix} 25 & 51 & 78 \\ 17 & 16 & 10 \\ 35 & 65 & 98 \\ 82 & 61 & 91 \end{vmatrix}$$

T( ) = 17 T( ) = 91 T( ) = 61

T( ) = 65 T( ) = 78 T( ) = 16

T( ) = 25 T( ) = 82 T( ) = 10

A DIMension statement must include the number of rows and the number of columns to reserve space for a matrix.

$$A = \begin{vmatrix} 3 & 5 \\ 0 & 2 \\ 1 & 1 \end{vmatrix}$$

DIM A(3,2)

3. Write an appropriate DIM statement for each array.

10 DIM _____

$$R = \begin{vmatrix} 13 & 45 & 23 \\ 20 & 15 & 30 \\ 12 & 19 & 98 \end{vmatrix}$$

_____

$$S = \begin{vmatrix} 9 & 15 & 89 & 0 & 1 & 0 \\ 19 & 35 & 90 & 9 & 6 & 9 \end{vmatrix}$$

_____

$$T = \begin{vmatrix} 7 & 32 & 15 & 99 & 9 \\ 55 & 75 & 8 & 22 & 12 \\ 0 & 1 & 1 & 10 & 1 \\ 12 & 45 & 85 & 5 & 6 \end{vmatrix}$$

_____

$$N\$ = \begin{vmatrix} JOE & RONDA & GREG & JANET \\ SUE & JACK & JIM & JAMES \\ KAY & JOSEPH & NANCY & STACY \\ JOHN & SUSAN & TOM & BONNIE \\ KATHY & JULIE & LEE & DIANA \end{vmatrix}$$

4. Insert missing statements or complete the incomplete statements in each program which will store and print the above matrices.

```
10
20  FOR I =   TO
30  FOR J =   TO
40  READ R(I,J)
50  PRINT
60  NEXT
70  PRINT
80  NEXT I
90  DATA 13, .....
95  DATA 20, .....
98  DATA 12, .....
99  END
```

```
10
20
30  FOR   = 1 TO 6
40
50  PRINT S
60  NEXT J
70  PRINT
80  NEXT I
90  DATA 9, 15,  ...
95  DATA 19, 35, ...
99  END
```

```
10
20  FOR J = 1 TO
30  FOR I = 1 TO
40  READ T(I)
50  PRINT T(J,I)
60  NEXT I
70  PRINT
80  NEXT J
90  DATA 7, 32, .....
95  DATA 55, .....
96  DATA 0, 1, .....
97  DATA 12, .....
99  END
```

5. Each of the following programs needs additional statements to be complete. Study the program carefully before inserting statements.

```
05  DIM T(4,6)
10  REM 4X6 MATRIX
20  REM ADDING COLUMN 4
30  FOR I = 1 TO 4
40  FOR J = 1 to 6
50  READ T(I,J)
60  NEXT J
70  NEXT I
75  FOR I = 1 TO 4
80  LET J = 4
85
90  NEXT I
95  PRINT S " IS THE SUM";
96  PRINT " OF THE 4TH COLUMN."
99  END
```

```
05  DIM H(2,3), G(2,3), V(2,3)
10  REM ADDING TWO MATRICES
20  FOR I = 1 TO 2
30  FOR J = 1 to 3
40  READ H(I,J)
50  READ G(I,J)
60
70  PRINT V(I,J),
75  NEXT J
80  PRINT
85  NEXT I
90  DATA 2,4,.............
95  END
```

```
010 DIM A(4,5), B(4,5) S,(4,5)
020 REM WORKING WITH MATRICES
030 FOR I = 1 TO 4
040 FOR J = 1 TO 5
050
060 READ B(I,J)
070 NEXT J
075 NEXT I
080 REM MULTIPLYING MATRICE A
081 REM BY 10
090 FOR I = 1 TO 4
095 FOR J = 1 TO 5
099
100 PRINT S(I,J),
110 NEXT J
115 PRINT
120 NEXT I
125 REM ADDING MATRIX A
130 FOR I = 1 TO 4
135 FOR J = 1 to 5
140
145 PRINT M " IS THE SUM."
```

```
150 NEXT J
155 NEXT
160 REM ADDING ROW 2 OF MATRIX A
165
170 LET I = 2
175 LET D = D + A(I,J)
180 NEXT J
185 PRINT "THE SUM OF ROW 2 IS " D
190 REM FINDING A SPECIAL VALUE IN B
195 FOR I = 1 TO 4
200 FOR J = 1 TO 5
210 IF B(I,J) = 16 THEN 230
220
230 PRINT "SIXTEEN IS IN MATRIX B."
240 NEXT I
220 NEXT J
300 DATA 14, 18, 23,.....
310 END
```

Assume each of the following matrices have been properly stored in the computer's memory. Insert appropriate statements to accomplish the task described.

$$A = \begin{vmatrix} 5 & 6 & 8 \\ 2 & 1 & 0 \end{vmatrix} \quad B = \begin{vmatrix} 6 & 7 \\ 0 & 0 \\ 2 & 3 \end{vmatrix} \quad C = \begin{vmatrix} 15 & 35 & 95 \\ 20 & 25 & 30 \\ 65 & 70 & 40 \end{vmatrix} \quad D = \begin{vmatrix} 1 & 3 & 5 \\ 6 & 4 & 2 \\ 0 & 7 & -9 \end{vmatrix}$$

6. Search matrix D for negative values and change them to positive values.

```
100  REM NEGATIVE TO POSITIVE
110  FOR I = 1 TO 2
120  FOR J = 1 TO 3
130
140  GOTO 160
150  LET D(I,J) = -1 * D(I,J)
160  NEXT J
170  NEXT I
```

7. Form a new matrix E by subtracting matrix D from matrix C.

```
200  REM NEW MATRIX E
205  DIM E(3,3)
210  FOR I = 1 TO 3
220  FOR J = 1 TO 3
230
240  NEXT J
250  NEXT I
```

8. Change all the values in row two in matrix C to zero.

```
300  REM  ROW 2 TO ZERO
310  FOR J = 1 TO 3
320
330
340  NEXT J
```

9. Count all of the elements equal to zero in matrix B.

```
400  REM COUNTING ZEROS
410  FOR I = 1 TO 3
420  FOR J = 1 TO 2
430  IF B(I,J) = 0 THEN 450
440  GOTO 460
450
460  NEXT J
470  NEXT I
```

10. Compare the elements in matrix C to matrix D and print any elements which are contained in both.

```
500  REM COMPARING C AND D
505  REM PRINT COMMON ELEMENTS
510
520
530  IF C(I,J) = D(I,J) THEN 550
540  GOTO 560
550  PRINT C(I,J);" IS IN D AND C"
560
```

11. Add rows 1 and 2 in matrix A to form array G.

```
600  REM SUM OF ROWS 1 AND 2
610  LET I = 1
620  FOR J = 1 TO 3
630
640  PRINT E(J)
650  NEXT J
```

12. Add columns 1 and 2 in matrix D to form matrix H(3,2).

```
700  REM FORM MATRIX H
710
720  FOR J = 1 TO 3
730
740  NEXT J
750  FOR I = 1 TO 3
760  FOR J = 1 TO 2
770  PRINT H(3,2);
780  NEXT J
790
800  NEXT I
```

---

*From Everybody's BASIC by Kay Richardson, pages 65-68; copyright (c) 1983 by Meka Publishing Co., Indianapolis, Indiana. Reproduced with permission.

13. Design and code a program that will generate the following output. Use Data statements provided below.

Number Items Sold

NAME	JAN	FEB	MAR	*TOTAL*
Company A	500	350	725	
Company B	468	410	750	
Company C	389	310	500	
Company D	206	600	463	
TOTAL				* *

```
DATA  A,B,C,D
DATA  500,350,725
DATA  468,410,750
DATA  389,310,500
DATA  206,600,463
```

NAME _____
DATE _____
PERIOD _____

TEST ON ARRAYS*

1. Determine whether each of the following statements is valid or invalid. If invalid, state the syntax rule violated.

a. 100 A(I) = I
b. 100 I = A(I)
c. 100 A = A(I)
d. 100 IF B3(2) = B2(3) THEN 100
e. 100 DIM A(12)
   110 PRINT A
f. 100 I = -3
   110 A(I) = 2
g. 100 A$(2) = "BILLY"
h. 100 A$(2) = B$
i. 100 A(I+J) = A(I) + A(J)
j. 100 A(I) + A(J) = A(I+J)
k. 100 A(K) = A(I*J)

2. What would be stored in array A as a result of the following program?

```
100 FOR K = 1 TO 5
110 A(K) = K*2+1
120 NEXT K
```

3. What would be printed by this program?

```
100 DIM A(5), N$(5)
110 FOR K = 1 TO 5
120   READ A(K)
130 NEXT K
140 FOR K = 1 TO 5
150   READ N$(K)
160 NEXT K
170 FOR K = 1 TO 5
180   PRINT A(K), N$(K)
190 NEXT K
200 DATA 8,2,0,8,43,BEN,JEN,SAM,
    JESS,ALLIE
210 END
```

4. Why would this program produce an error message?

```
100 DIM A(5), N$(5)
110 FOR K = 5 TO 1 STEP -1
120   READ A(K), N$(K)
130 NEXT K
140 FOR K = 1 TO 5
150   PRINT N$(K), A(K)
160 NEXT K
170 DATA 8,2,0,8,43,BEN,JEN,SAM,
    JESS,ALLIE
180 END
```

5. What will be printed by this program?

```
100 DIM A(10), B(10)
110 FOR K = 1 TO 10 STEP 2
120   READ A(K)
130 NEXT K
140 FOR K = 2 TO 10 STEP 3
150   READ B(K)
160 NEXT K
165 I = 0
170 FOR K = 1 TO 10
180   IF A(K) = B(K) THEN 220
190     TS = B(K)
200     B(K) = A(K)
210     A(K) = TS
215     I = I + 1
220 NEXT K
230 FOR K = 1 TO 10
240   PRINT A(K), B(K)
250 NEXT K
260 PRINT "THE NUMBER OF INTER-
    CHANGES IS ";I
270 DATA 3, -9, 2, 8, 1, 4, 2, 3
280 END
```

6. Given array A and variable I and J with the following values stored in memory:

A(1)	A(2)	A(3)	A(4)	A(5)	A(6)	I	J
4	7.2	0	-1.1	88	-12.3	2	3

Evaluate each of the following:

a. A(I)
b. A(I^2)
c. A(I+3)
d. A(3) + A(I)
e. A(I) * A(I)
f. 2 * A(2)
g. A(I)/A(1) * A(J)
h. ABS(INT(A(4)))
i. SQR(A(1))
j. J * A(J)
k. A(J-I)
l. A(J) - A(I)

7. In reference to the above array A and variables I and J, identify each of the following as valid or invalid.

a. -1 * A
b. SQR(A(J))
c. A(A(1))
d. A(I*5)
e. 5A(I)

8. Given a two-dimensional array S which is 5 by 5, write a line or lines of code that will display the:

a. Fourth row of S
b. First column of S
c. Rows 2 through 4
d. Columns 1 and 5
e. Main diagonal of S
f. The element in the center of S

9. Given array M and variables I and J with the following values stored in memory:

M(1)	M(2)	M(3)	M(4)	M(5)	M(6)	M(7)	M(8)	I	J
3	4	1.0	2	12	8	-100	6	3	?

What would be the output of the following program?

```
•
•
•
100 M(5) = M(I) + M(J)
110 M(I) = M(I) + M(J)
120 M(6) = M(I+J)
130 M(7) = M(I*J)
140 M(J) = M(I)
150 M(I) = M(J)
160 M(I-J) = 22
170 M(I*J) = 0
180 FOR K = 1 TO 8
190   PRINT M(K)
200 NEXT K
```

*From Programming in Apple BASIC by John J. Dielsi, Elaine S. Grossman, John P. Tucciarone, pages 253-255; copyright (c) 1984 by CBS College Publishing, New York, New York. Reproduced with permission.

Handout #12
Variables and Strategies

NAME_____
DATE_____
PERIOD_____

ERROR-CHECKING SUBROUTINE

```
010  REM ***MAIN PROGRAM***
020  HOME
030  INPUT "LOAN AMOUNT"; L
040  S = L : REM LOADING SUBROUTINE NUMBER
050  GOSUB 150
060  INPUT "INTEREST RATE"; I
070  S = I : REM LOADING SUBROUTINE NUMBER
080  GOSUB 150
090  INPUT "YEARS TO PAY"; Y
100  S = Y : REM LOADING SUBROUTINE NUMBER
110  GOSUB 150
120  REM ***ALL ENTRIES CHECKED...OUTPUT FOLLOWS***
130  PRINT "ALL NUMBERS ACCEPTED"
140  GOTO 260
150  REM ***SUBROUTINE FOR ENTRY CHECK***
160  IF S > 0 THEN 250
170  HOME
180  FOR C = 1 TO 10
190     PRINT "        ILLEGAL ENTRY...START OVER!"
200     PRINT
210     FOR D = 1 TO 50
220     NEXT D : REM DELAY LOOP
230  NEXT C
240  FOR D = 1 TO 500 : NEXT D : RUN
250  RETURN
260  END
```

DEBUGGING EXERCISE*

HOTEL CHARGE PROGRAM

DIRECTIONS: The program below contains errors. Locate the lines causing the errors and correct (DEBUG) them in the space provided.

C = cost D = days N$ = name R$ = rate(class A or B)

```
010  REM***HOTEL CHARGE PROGRAM***
020  PRINTTAB(7); "HOTEL RESIDENTS"
030  PRINT
040  PRINTTAB(5); "NAME"; TAB(17); "DAYS";
     TAB(23); "CHARGE"
050  PR..
060  READ N$, D, R
070  IF N$ = "END OF DATA" THEN 230
080  IF R = A THEN 110
090  IF R = B THEN 140
100  PRINT ERROR IN DATA OF"; N
110  GOTO 60
120  C = D * 40
130  GOTO 150
140  C = D * 50
150  PRINT N$; TAB17: D; TAB24: "$"; C
160  GOTO 60
170  REM***  ***DATA*******
180  DATA BILL WILLIAMS, A, 3
190  DATA DONNA STANLEY, 2, B
200  DATA BOB FRENCH, A, 1
210  DATA VIRGINIA OWEN, 4, B
220  DATA END OF DATA, 0, 0
230  PRINT "END OF PROCESSING"
240  END
```

When the above program is DEBUGGED it will produce the following output:

HOTEL RESIDENTS

NAME	DAYS	CHARGE
BILL WILLIAMS	3	$120
DONNA STANLEY	2	$100
BOB FRENCH	1	$ 40
VIRGINIA OWEN	4	$200
END OF PROCESSING		

*Further reproduction is prohibited. See footnote on page 77.

DEBUGGING EXERCISE*

AIRPLANE FLIGHTS

DIRECTIONS: The program below contains errors. Locate the lines causing the errors and correct (DEBUG) them in the space provided.

C$ = Chicago F$ = Flight M$ = Memphis N$ = New York

```
010  REM THIS PROGRAM IDENTIFIES SELECTED
020  REM FLIGHTS LEAVING MEMPHIS,
030  REM CHICAGO, AND NEW YORK
040  PRINT "ENTER CITY INITIAL TO DISPLAY
     FLIGHT NUMBER AND DEPARTURE TIME
050  PRINT "C = CHICAGO"
060  PRINT "M = MEMPHIS"
070  PRINT "N = NEW YORK"
080  PRINT "ENTER C, M, OR N FOR FLIGHT
     NUMBER AND DEPARTURE TIME"
090  OUTPUT F$
100  IF F$ = "C$" THEN 150
110  IF F$ = "M$" THEN 190
120  IF F$ = "N$" THEN 230
130  PRINT "PLEASE TRY AGAIN"
140  GOT 50
150  PRINT "FLIGHT 445 -- DEP. 10:05A
160  PRINT "FLIGHT 427 -- DEP. 2:15P
170  PRINT FLIGHT 458 -- DEP. 3:15P"
180  GOTO 250
190  PRINT FLIGHT 442 -- DEP. 7:15A"
200  PRINT "FLIGHT 492 -- DEP. 730A"
210  PRINT "FLIGHT 408 -- DEP. 4:05P"
220  GOTO 250
230  PRINT "FLIGHT 405 -- DEP 10:52A
240  PRINT "FLIGHT 409 -- DEP 10:25A
250  PRINT
260  PRINT "END OF SCHEDULE"
270  END
```

When the above program is DEBUGGED it will produce the followin output:

```
ENTER CITY INITIAL TO DISPLAY FLIGHT NUMBER
AND DEPARTURE TIME
C = CHICAGO
M = MEMPHIS
N = NEW YORK
ENTER C, M, OR N FOR FLIGHT NUMBER AND
DEPARTURE TIME
?
```

(If a numeric character or an alphabetic character other than C, M, or N is entered, the following prompt will be displayed--followed by the output displayed above.)

PLEASE TRY AGAIN

*Further reproduction is prohibited. See footnote on page 77.

DEBUGGING EXERCISE*

UTILITY BILLING LIST

DIRECTIONS: The program below contains errors. Locate the lines causing the errors and correct (DEBUG) them in the space provided.

AD = amount due PB = previous balance TB = total billing this period
CN = customer number T = total customers TD = total delinquent amount

```
010  REM UTILITY BILLING LIST
020  T = 0: TD = 0: TB = 0
030  READ CN, PB, A
040  PRINT
050  IF CN = 0 THEN 90
060  AD = A
070  IF PB = 0 THEN 130
080  AD = AD + PB
090  TO = TO + PB
100  PRINT CN: " AMOUNT DUE": AD;
     " AMOUNT OVERDUE"; PB
110  T = T + 1
120  TB TB AD
130  GOTO 30
140  PRINT
150  PRINT T: "CUSTOMERS PROCESSED"
160  PRINT TOTAL BILLING THIS PERIOD = ";
     TB
170  PRINT TOTAL DELINQUENT AMOUNT = ";
     TD
180  PRINT "END OF REPORT"
190  DATA 101,0.00,34.52,107,0.00,32.69
200  DATA 134,0.00,29.80,152,31.79,33.41
210  END
```

When the above program is DEBUGGED it will produce the following output:

```
101 AMOUNT DUE 34.52
AMOUNT OVERDUE 0

107 AMOUNT DUE 32.69
AMOUNT OVERDUE 0

134 AMOUNT DUE 29.80
AMOUNT OVERDUE 0

152 AMOUNT DUE 65.20
AMOUNT OVERDUE 31.79
```

4 CUSTOMERS PROCESSED
TOTAL BILLING THIS PERIOD = 162.21

TOTAL DELINQUENT AMOUNT = 31.79
END OF REPORT

DEBUGGING EXERCISE*

CABLE TV BILLING

Directions: The program below contains errors. Locate the lines causing the errors and correct (DEBUG) them in the space provided. The local TV cable company charges $15.00 for regular hookup and HBO and an additional $7.00 for Cinemax, Source, and "Special Offerings."

A = amount of service S$ = service (H/S) TA = total amount of billing
CN = customer number T = total customers TH = total HBO customers

```
010  REM CABLE TV BILLING
020  T = 0; TA = 0; TH = 0
030  READ CN, S$
040  IF CN = 0 THEN 130
050  T = T + 1
060  A = 7.00
070  IF S$ = "S" THEN 100
080  TH = TH + 1
090  A = A + 15.00
100  TA = TA + A
110  PRINT CN: "AMOUNT DUE"; A
120  GOTO 40
130  PRINT
140  PRINT T: "CUSTOMERS PROCESSED"
150  PRINT "TOTAL BILLING ="; TA
160  PRINT "NUMBER HBO CUSTOMERS ="; TH
170  PRINT "END OF REPORT"
180  DATA 1112, S, 1213, H, 1310, S
190  DATA 1401, H, 1479  H, 1523, S
200  DATA 1577, H, 1603, S; 1776, H.
210  DATA 0, 0.
220  END
```

*Further reproduction is prohibited. See footnote on page 77.

When the above program is DEBUGGED it will produce the following output:

```
1112   AMOUNT DUE 15
1213   AMOUNT DUE 22
1310   AMOUNT DUE 15
1401   AMOUNT DUE 22
1479   AMOUNT DUE 22
1523   AMOUNT DUE 15
1577   AMOUNT DUE 22
1603   AMOUNT DUE 15
1776   AMOUNT DUE 22

9 CUSTOMERS PROCESSED
TOTAL BILLING = 170
NUMBER HBO CUSTOMERS = 5
END OF REPORT
```

*From a set of 50 masters and 74 visual masters, Micro 5: Reviewing and Debugging, copyright (c) 1984. Full sets available from the publisher, J. Walch, Publisher, Portland, Maine 04104-0658. Used by permission. Further reproduction is prohibited.

NAME_____
DATE_____
PERIOD_____

MAT-READ, MAT-PRINT, MAT-INPUT STATEMENTS*

PROGRAM:

```
10  REM  PROGRAM READS AND PRINTS A 5X8 MATRIX
20  DIM M(5,8)
30  MAT READ M
40  MAT PRINT M;
50  DATA 45,58,59,75,74,76,86,94,13,12,41,63,52,20,37,29,18,24,16,35
60  DATA 30,56,31,24,25,65,64,98,87,82,81,73,50,60,86,94,83,41,62,31
70  END
```

OUTPUT:

45	58	59	75	74	76	86	94
13	12	41	63	52	20	37	29
18	24	16	35	30	56	31	24
25	65	64	98	87	82	81	73
50	60	86	94	83	41	62	31

PROGRAM:

```
010  REM  PROGRAM ADDS ROWS AND ELEMENTS OF A 3X3 INPUT MATRIX
030  DIM M(3,3)
045  PRINT "TYPE IN THE MATRIX AFTER THE QUESTION MARK"
050  MAT INPUT M
055  PRINT
060  T=0
070  FOR I = 1 TO 3
080  R(I)=0
090  FOR J = 1 TO 3
100  R(I)=R(I)+M(I,J)
110  NEXT J
120  T=T+R(1)
130  NEXT I
140  MAT PRINT M;
145  PRINT : PRINT
```

```
150  FOR I= 1 TO 3
160  PRINT "THE TOTAL FOR ROW ";I;" IS ";R(I)
170  NEXT I
180  PRINT "THE TOTAL OF ALL ELEMENTS IS ";T
220  END
```

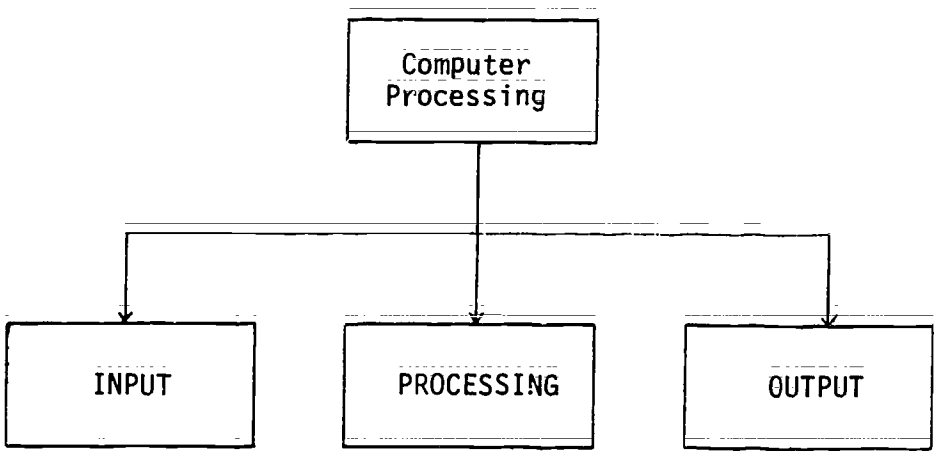
OUTPUT:

```
TYPE IN THE MATRIX AFTER QUESTION MARK
? 1,2,3,4,5,6,7,8,9

 1   2   3

 4   5   6

 7   8   9

THE TOTAL FOR ROW 1 IS 6
THE TOTAL FOR ROW 2 IS 15
THE TOTAL FOR ROW 3 IS 24
THE TOTAL OF ALL ELEMENTS IS 45
```

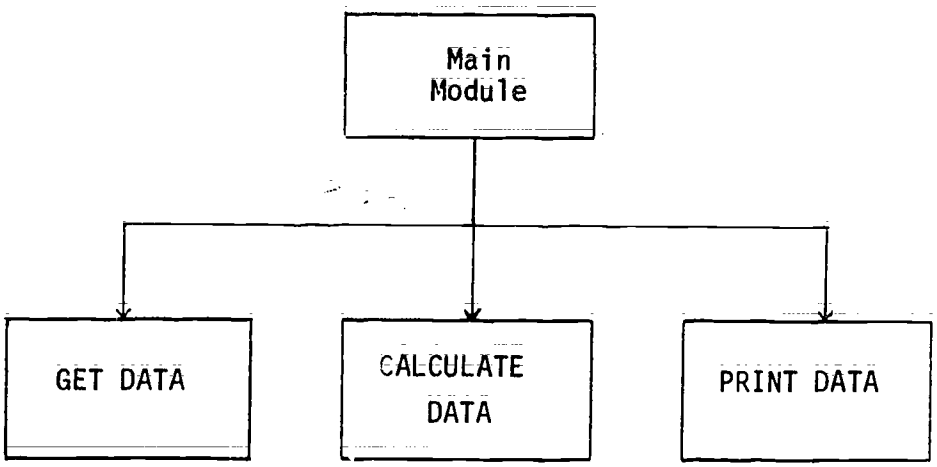
*From The BASICS of BASIC by Alfredo C. Gomez, pages 192, 194; copyright (c) 1983 by CBS College Publishing, New York, New York. Reproduced with permission.

HIERARCHY CHART

Basic Computer Processing Steps:



Basic Hierarchy Chart:



NAME _____
DATE _____
PERIOD _____

DESIGNING HIERARCHY CHARTS and PROGRAMS*

Program 1

Many employees are paid on an hourly basis. That is, the amount of pay is determined by multiplying the number of hours worked times the hourly rate of pay. The product of this multiplication is called gross pay, the amount of pay before any deductions are made for income tax, Social Security (FICA), etc. Write a program to calculate and print gross pay. The program should clear the screen prior to beginning the printout. It should then print each employee's name, number of hours worked, pay rate, and gross pay. The data to be used by the program should be on DATA lines. The first item is the employee's name, followed by the number of hours worked and the pay rate. Here are the data lines:

```
5000 DATA "ABLE, MARTHA", 40, 5.20
5010 DATA "CARDWELL, HELEN", 38, 4.83
5020 DATA "MIMMS, FRED", 35.5, 3.97
5030 DATA "SMITH, MARILYN", 40, 3.97
5040 DATA "WILLIAMS, JAMES", 38, 5.20
5050 DATA "EOD", 0, 0
```

Make the output as neat and easy to read as possible. Use an appropriate main heading and column headings, with the column headings aligned. The submodules used in the program might be Get Data, Calculate, and Print Payroll.

Program 2

Frequently employees are paid 1½ times their regular rate for any hour worked beyond 40 each week. These hours beyond 40 are referred to as overtime. Write a program similar to Program 1 (or modify Program 1) so that employees receive their regular pay for the first 40 hours and 1½ times their regular rate for hours over 40. The printout should contain columns for employee name, number of regular hours worked, number of overtime hours worked, regular pay rate, amount of regular gross pay, amount of overtime gross pay, and total gross pay (regular and overtime added together). The data lines to be used are:

```
8000 DATA "BURT, WILSON", 40, 4.21
8010 DATA "CALMER, HELEN", 43, 3.90
8020 DATA "JONES, SARAH", 42.75, 5.45
8030 DATA "KEITH, WILLIAM", 38, 5.20
8040 DATA "MICHAELS, FRANK", 49, 4.80
8050 DATA "EOD", 0, 0
```

Program 3 — Modify Program 2 so that the number of hours worked is input from the keyboard rather than being read from data lines. Include an error trap in the program to ensure against accidental entry of too many hours. Company policy prohibits any employee from working more than 50 hours per week.

Enrichment Program — Many businesses have salaried employees and commissioned employees in addition to hourly employees. Salaried employees receive the same amount of gross pay regardless of the number of hours they work. Commissioned employees receive a percentage of their sales as pay. A program to figure the payroll for such a business needs to be able to handle all the types of pay calculation. In other words, there should be a separate submodule for calculation of each type of gross pay, with the appropriate one being called by the main module.

Write such a program. Use data lines to hold constant data, such as name, pay type, and rate. Enter variable data, such as hours worked or amount of sales from the keyboard when the program is run. Here are the data. They are not arranged properly for data lines.

Employee Name	Type	Rate	Suggested Input
Copenhaver, W.	Hourly	4.97	43 hours
Croan, P.	Salary	250.00	
Haley, B.	Salary	263.00	
Ivanoski, G.	Commission	9%	$4000 in sales
Jump, R.	Hourly	5.90	40 hours
Laseur, K.	Hourly	4.50	41 hours

*STRUCTURED BASIC by Clark and Drum, Chapter 8, pp. 131-132; copyright (c) 1983 by South-Western Publishing Company, Cincinnati, Ohio. Reproduced with permission.

NAME_____
DATE_____
PERIOD_____

MENU-DRIVEN PROGRAMMING ASSIGNMENTS*

Program 1 Bookstore listing of books are to be prepared utilizing the following MENU:

BOOKSTORE MENU

CODE	FUNCTION
1	Enter Book Information
2	List Books by Title
3	List Books by Author
4	End Program

ENTER 1, 2, 3 or 4 TO MAKE SELECTION:

For CODE 1 the following screen should be developed using six books to be entered by the user.

ENTER BOOK INFORMATION

BOOK NUMBER 1

Book Title:
Book Author:
Quantity:
Book Price:

(DATA lines with READ statements may be used while developing and testing the program. INPUT statements must be used for the finished program.)

For CODE 2 the following report should be displayed:

BOOK REPORT BY TITLE

Title	Author	Qty	Price
Fast Water	Numovich	8	12.95
Sea and Stone	Allergen	21	14.95
Shock Light	Brannigan	14	15.95
The Marchers	Iotay	8	10.95
Torch Gas	Critener	12	12.95
Winds of Time	Pollutey	4	12.95

Total Quantity: 67
Total Value: $ 935.65

DEPRESS ENTER OR RETURN KEY TO RETURN TO THE MENU:

For CODE 3 the following report should be displayed:

BOOK REPORT BY AUTHOR

Author	Title	Qty	Price
Allergen	Sea and Stone	8	12.95
Brannigan	Shock Light	21	14.95
Critener	Torch Gas	14	15.95
Iotay	The Marchers	8	10.95
Numovich	Fast Water	12	12.95
Pollutey	Winds of Time	4	12.95

DEPRESS ENTER OR RETURN KEY TO RETURN TO THE MENU:

Program 2 Class listings for a school are to be prepared. A program should be designed and coded to produce the following listings:

CLASS MENU

CODE FUNCTION

1 Enter Class Information
2 Display in Class Name Sequence
3 Display in Teacher Name Sequence
4 Obtain Teacher Name and Enrollment
5 End Program

ENTER 1, 2, 3, 4 or 5 TO MAKE SELECTION:

For CODE 1 the following screen should be developed.

ENTER CLASS INFORMATION

CLASS NUMBER 1

Enter Class Name:
Enter Teacher Name:
Enter Enrollment:

For CODE 2 the following screen should be developed.

DISPLAY CLASS NAMES IN SEQUENCE

Class	Teacher	Enroll.
Bus 231	Harrelson	128
Bus 429	Abbot	32

Che 112	Chemonte	359
Che 213	Zunderrey	21
Phy 101	Nommerrei	573
Soc 219	Berret	45

Total Enrollment: 1 [illegible]

DEPRESS ENTER OR RETURN KEY TO [illegible]RN TO THE MENU:

For CODE 3 the following screen should be developed:

DISPLAY TEACHER NAMES IN SEQUENCE

Teacher	Class Name	Enroll.
Abbott	Bus 429	32
Berret	Soc 219	45
Chemonte	Che 112	359
Harrelson	Bus 231	128
Nommerrei	Phy 101	573
Zunderrey	Che 213	21

Total Enrollment: 1,158

DEPRESS ENTER OR RETURN KEY TO RETURN TO THE MENU:

For CODE 4 the following screen should be developed:

OBTAIN TEACHER NAME AND ENROLLMENT

Enter Class Name:
Enter Teacher Name:
Enter Enrollment:

DEPRESS ENTER OR RETURN KEY TO RETURN TO THE MENU:

*INTRODUCTION TO BASIC PROGRAMMING by Shelly and Cashman, Chapter 8, pp. 8.58-8.61; copyright (c) 1982 by Anaheim Publishing Company, Brea, California. Reproduced with permission.

POSSIBLE SOLUTION TO MENU-DRIVEN PROGRAM 1

```
0100 REM THIS PROGRAM ENTERS BOOK INFORMATION, PRINTS A REPORT BY TITLES
0110 REM OR BY AUTHOR THROUGH THE USE OF A MENU.
0120 REM FOR THE TRS-80 MODEL 4 AND IBM PC COMPUTERS
0130 REM
0140 '
0150 REM VARIABLE NAMES USED
0160 '   BOOK.TITLE$()=BOOK TITLE ARRAY
0170 '   AUTHOR$()=NAME OF AUTHOR ARRAY
0180 '   QUANTITY()=QUANTITY ARRAY
0190 '   PRICE()=PRICE ARRAY
0200 '   N=NO. OF ITEMS IN ARRAY
0210 '   CHOICE=MENU SELECTION
0220 '   S=SUBSCRIPT FOR ARRAY
0230 '   TOT.QUANT=TOTAL QUANTITY
0240 '   TOT.PRICE=TOTAL PRICE
0250 '   GRAND.TOT=GRAND TOTAL
0290 '
0300 REM***INITIALIZATION OF VARIABLES***
0310 N=6
0320 F$ = "\      \    \              \     ##   ##.##"
0330 F1$ = "\        \     \              \     ##   ##.##"
0340 DIM BOOK.TITLE$(N), AUTHOR$(N), QUANTITY(N), PRICE(N)
0350 '
0500 REM***MAIN MODULE - DISPLAY MENU AND GET SELECTION***
0510 CLS
0520 PRINT: PRINT: PRINT:
0530 PRINT TAB(15) "BOOKSTORE MENU": PRINT
0540 PRINT TAB(10) "CODE" TAB(20) "FUNCTION": PRINT
0550 PRINT TAB(11) "1" TAB(15) "ENTER BOOK INFORMATION"
0560 PRINT TAB(11) "2" TAB(15) "LIST BOOKS BY TITLE"
0570 PRINT TAB(11) "3" TAB(15) "LIST BOOKS BY AUTHOR"
0580 PRINT TAB(11) "4" TAB(15) "END PROGRAM"
0590 PRINT
0600 INPUT "ENTER A NUMBER 1 THROUGH 4: "; CHOICE
0610 IF CHOICE>=1 AND CHOICE<=4 THEN 0660
0620 PRINT
0630 PRINT "   "; CHOICE; " IS INVALID"
0640 INPUT "     PLEASE RE-ENTER 1, 2, 3 OR 4: "; CHOICE
0650 GOTO 0610
0660 IF CHOICE=4 THEN 0690
0670 ON CHOICE GOSUB 1000, 2000, 3000
0680 GOTO 0500
0690 PRINT: PRINT "END OF BOOKSTORE PROGRAM"
0700 END
0710 '
1000 REM***ENTER BOOK INFORMATION***
1010 CLS
1020 FOR S=1 TO N
```

```
1030    PRINT "ENTERING BOOK INFORMATION": PRINT
1040    PRINT "BOOK NUMBER "; S: PRINT
1050    INPUT "   ENTER TITLE OF BOOK: "; BOOK.TITLE$(S): PRINT
1060    INPUT "   ENTER AUTHOR OF BOOK: "; AUTHOR$(S): PRINT
1070    INPUT "   ENTER QUANTITY: "; QUANTITY(S): PRINT
1080    INPUT "   ENTER PRICE OF BOOK: "; PRICE(S): PRINT
1085 NEXT S
1090 RETURN
1100 '
2000 REM***LIST BOOKS BY TITLES***
2010 CLS
2020 PRINT "     TITLE          AUTHOR        QTY     PRICE": PRINT
2025 TOT.QUANT=0 : TOT.PRICE=0 : GRAND.TOT=0
2030 FOR S=1 TO N
2040    PRINT USING F$; BOOK.TITLE$(S); AUTHOR$(S); QUANTITY(S); PRICE(S)
2050    TOT.QUANT=TOT.QUANT+QUANTITY(S)
2060    TOT.PRICE=QUANTITY(S) * PRICE(S)
2070    GRAND.TOT=GRAND.TOT+TOT.PRICE
2080 NEXT S
2090 PRINT: PRINT: PRINT "TOTAL QUANTITY:  "; TOT.QUANT
2100 PRINT: PRINT "TOTAL VALUE:     "; GRAND.TOT
2110 PRINT: PRINT "PRESS ENTER TO CONTINUE": PRINT
2120 INPUT "  RETURN TO MENU"; RESPONSE$
2130 RETURN
2140 '
3000    **LIST BOOKS BY AUTHOR***
3010 CLS
3020 REM***SORTING ROUTINE***
3030 PRINT "SORTING"
3040 FOR B1=1 TO N-1
3050    FOR B2=B1+1 TO N
3060        IF AUTHOR$(B2)>=AUTHOR$(B1) THEN GOTO 3110
3070        SWAP BOOK.TITLE$(B1), BOOK.TITLE$(B2)
3080        SWAP AUTHOR$(B1), AUTHOR$(B2)
3090        SWAP QUANTITY(B1), QUANTITY(B2)
3100        SWAP PRICE(B1), PRICE(B2)
3110    NEXT B2
3120 NEXT B1
3130 CLS
3140 PRINT "          BOOK REPORT BY AUTHOR": PRINT
3150 PRINT "  AUTHOR          TITLE          QTY    PRICE": PRINT
3160 FOR S=1 TO N
3170    PRINT USING F1$; AUTHOR$(S); BOOK.TITLE$(S); QUANTITY(S); PRICE(S)
3180 NEXT S
3190 PRINT: INPUT "PRESS ENTER TO CONTINUE"; RESPONSE$
3200 RETURN
3210 '
```

SAMPLE ACTIVITY CLUSTER #4

BASIC Functions and Graphics

Topics:

1. Intrinsic Mathematical Functions
2. Intrinsic String Manipulation Functions
3. User-Defined Functions
4. High-Resolution Graphics

Classroom Management:

Most of the work is best done by pairs of students. The culminating projects in functions and in graphics should be assigned as individual work.

Materials:

Demonstration microcomputer and large-screen monitor;
Handouts.

Time for Activities:

Approximately three to fou[illegible]uld be allowed, depending on the overall capability and int[illegible]els of the students.

Teacher Preparation and Procedures:

Prepare the necessary handouts. Experiment with some graphics problems to know which would be most appropriate for the students. All sample materials for these activities are located in the Sample Assignments and Materials for Activity Cluster #4.

1. Present a lecture on BASIC functions, using a demo microcomputer to show examples of intrinsic string functions.

2. Distribute Handout #1 - String Functions Worksheet for pairs of students to practice string functions. Intermingle with students in the lab to answer any questions.

3. Assign a string function problem to [illegible]ch of six groups of students from Handout #2 - String Functio[illegible] Group Problems. Have them prepare a flowchart first and discuss the most efficient means for solving the problem. After programs have been coded and tested, each group should share its results with the whole class, using a demo microcomputer and distributing listings of the program. Then from the same handout, have each student do one of the three short programs involving string functions. Emphasize the need for a flowchart and good documentation.

4. Complete a lecture on BASIC functions, discussing and demonstrating intrinsic mathematical functions and user-defined functions. Using Handout #3 - Intrinsic Math Functions Worksheet and Handout #4 - User-Defined Functions Worksheet, provide assignments for both mathematical and user-defined functions. Discuss results.

5. Have each student do two of five short programs involving user-defined functions from Handout #5 - User-Defined Functions Individual Problems. Require flowcharts and documentation.

6. Present a lecture on high-resolution graphics, demonstrating a few simple programs and distributing Handout #6 - High-Resolution Graphics Summary Sheet and Design Grid. Handout #7 - Sample Demo Hi-Res Graphics Problems provides three simple problems for demonstrating high-resolution graphics and one more complex problem that uses high-resolution graphics to graph polynomials of degree less than six. A sample listing is included as a possible solution to this latter problem. Point out the use of the IF-THEN and FOR-NEXT control structures, single-dimension arrays and subscripted variables, and subroutines.

7. Have pairs of students do the required graphics problems, while individuals may do the optional problems. Discuss and compare results, allowing students to run their programs for the whole class. Assign individuals to one of several longer graphics programs, according to their level of skills. After the due date has passed, have students share their results. The following handouts may be helpful: Handout #8 - Hi-Res Graphics Problems and Handout #9 - Hi-Res Graphics Individual Programs.

Sample Assignments and Materials for Activity Cluster #4:

The following pages in this activity cluster provide handouts that could be used in this course.

90

NAME_____
DATE_____
PERIOD_____

STRING FUNCTIONS WORKSHEET*

1. Run the following program to see how RIGHT$, LEFT$ and MID$ functions work:

```
5   HOME
10  B$ = "SPACESARENOTINCLUDEDHERE"
20  PRINT RIGHT$ (B$,4)
30  PRINT MID$ (B$,7,3)
40  PRINT LEFT$ (B$,6)
50  END
```

2. Include the following lines in the above program. Predict the output before running the program.

```
15  FOR N = 1 TO LEN (B$)
20  PRINT LEFT$ (B$,N)
30  PRINT RIGHT$ (B$,N)
40  PRINT MID$ (B$,1,25-N):PRINT
45  NEXT N
```

3. Before running the following program, predict the output.

```
5   HOME
10  X$ = "SUPERCALIFRAGILISTICEXPIALIDOCIOUS"
15  N = LEN (X$)
20  PRINT X$; " HAS":PRINT
25  PRINT N; " LETTERS OR CHARACTERS.":PRINT
30  PRINT "TYPE A NUMBER FROM 1 TO "; N:PRINT
35  PRINT "AND I WILL NAME THE CHARACTER":PRINT
40  PRINT "THAT IS IN THAT POSITION.":PRINT
45  PRINT "AN INVALID NUMBER STOPS THE RUN.": PRINT
50  INPUT "WHAT IS YOUR NUMBER ";A:PRINT
55  IF A<1 OR A>N GOTO 75
60  IF INT(A)<>A GOTO 75
65  HOME:PRINT MID$(X$,A,1);"  IS IN POSITION ";A:PRINT
70  GOTO 50
75  PRINT "INVALID NUMBER!":END
```

4. The following program uses the MI[illegible] LEN functions to decode a string. Predict the output before [illegible] it.

```
10  HOME
20  X$ = "EAVLELR YC OGWOSO DE ABTO YG RDAOSESS  OFRI NHEAIY !"
30  N = LEN(X$)
40  FOR [illegible] TO 2
50      [illegible] = S TO N STEP 2
60      [illegible] MID$ (X$,A,1);
70      [illegible]
100 PRINT:PRINT
110 NEXT S
120 END
```

5. Run the following program:

```
10  FOR J = 1 TO 7
15  READ X$(J)
20  NEXT J
25  DATA "MONDAY","TUESDAY","WEDNESDAY"
30  DATA "THURSDAY","FRIDAY","SATURDAY","SUNDAY"
35  PRINT "TYPE THE FIRST LETTER OF ANY":PRINT
40  PRINT "DAY AND THE COMPUTER WILL":PRINT
45  PRINT "NAME THAT DAY.":PRINT
50  INPUT "WHAT LETTER DO YOU CHOOSE  ?  ";D$:PRINT
60  FOR J = 1 TO 7
70  IF MID$(X$(J),1,1,)<=>MID$(D$,1,1)GOTO 90
80  PRINT X$(J):"  BEGINS WITH A  ";D$:PRINT
90  NEXT J
95  END
```

Change the program to:

a. Include a loop that gives the user a choice to continue selecting letters or to stop. When a character that does not represent a day of the week is picked, display an appropriate error message and stop the run.

b. Print the day which contains the second letter that the user picks (i.e., R=FRIDAY).

c. Print the name of the days that contain the most letters.

NAME _____
DATE _____
PERIOD _____

STRING FUNCTIONS GROUP PROBLEMS

1. Enter a string A$ and use a loop to print the ASCII number of each of its characters.*

2. Using the properly selected ASCII numbers in a DATA statement, print name of our high school.

3. Enter the string THREE!@#$%STRING!@#$%FUNCTIONS. Use LEFT$, MID$ and RIGHT$ to print the phrase THREE STRING FUNCTIONS.*

4. Enter a string of any length and print the length and the ASCII number of its first and last characters.*

5. Enter a string A$ and have the computer print the word with all the letter E's removed. Or input a string and output it without any vowels. Replace the vowels with a dash.

6. Write a program to accept a character string as input, and output the string with each character printed twice. For example DOUBLE would appear as DDOOUUBBLLEE.**

STRING FUNCTIONS INDIVIDUAL PROBLEMS

Do <u>One</u> of the Following Programs

A. Write a program to accept a message as a character string and encode the message. Construct the code as follows: Convert each character in the message to ASCII code, add 3, and convert the new ASCII code to its associated character. Then modify the program so that the user will have the option of encoding or decoding a message.**

---

*From <u>A Guide to Programming in Applesoft</u> by Bruce Presley, page 8.9; copyright (c) 1984 by Lawrenceville Press, Lawrenceville, New Jersey. Reproduced with permission.

**From <u>Programming in Apple BASIC</u> by John H. Dielsi, Elaine S. Grossman, John P. Tucciarone, pages 277-278; copyright (c) 1984 by CBS College Publishing, New York, New York. Reproduced with permission.

B. Enter a positive integer N$ as it would be expressed in binary form. Have the computer print the equivalent in the decimal system.*

C. Write a program to accept an extended message in sentence form and have the computer count the number of words in the sentence.

*From A Guide to Programming in Applesoft by Bruce Presley, page 8.9;

Handout #3
Functions and Graphics

NAME _____
DATE _____
PERIOD _____

INTRINSIC MATH FUNCTIONS WORKSHEET*

Determine the output:

1.
```
10   A = 7C.4
20   PRINT ABS(A),
3C   PRINT SG:(A)
4C   D
```

2.
```
10   A = 68.7
20   B = 4.7
30   PRINT SQR(A - B),
40   PRINT INT(A)
50   END
```

3.
```
10   PRINT INT(RND(1))
20   END
```

4.
```
10   PRINT SQR(INT((10+
     RND(1) + 40)/10))
20   END
```

5.
```
10   PRINT INT(SQR(40))
20   END
```

6.
```
10   R = INT(9*RND(1)) + 135
20   PRINT INT(R/9)
30   END
```

7.
```
10   FOR N = 1 TO 5
20   READ X,Y
3C   IF X/Y<=>INT(X/Y) GOTO 50
40   PRINT "OK"
50   NEXT N
60   DATA 7.3,30,4,33,11,3,6,8,1
70   END
```

8.
```
10   READ N
20   IF N/2 = INT(N/2) GOTO 40
30   GOTO 10
40   PRINT N
50   DATA 13,20,2,86,99,0
60   END
```

9.
```
10   READ X
2C   IF X = 0 GOTO 60
30   PRINT INT(X + 0.50),
40   GOTO 10
50   DATA 6.31, 84.71, 1C2.4, 0
60   END
```

10.
```
10   A = 3.3125
20   IF A > 50 GOTO 60
30   A = 2*A
40   PRINT INT(A + 0.5),
50   GOTO 20
60   END
```

11.
```
10   FOR X = 1 TO 3
20   R = INT(17*RND(1)) + 55
30   IF R/9<=>INT(R/9) GOTO 20
40   PRINT R,
50   NEXT X
60   END
```

12.
```
10   N = INT(45*RND(1)) + 4
20   PRINT INT((2*N +1)/N+2 +1)
30   END
```

*From Duplicating Masters -- Experiencing BASIC by Michael Mulcahy, page 11; copyright (c) 1984 by Media Materials, Inc., Baltimore, Maryland. Reproduced with permission.

Handout #4
Functions and Graphics

NAME _____
DATE _____
PERIOD _____

USER-DEFINED FUNCTIONS WORKSHEET*

Determine the output for this program

1.
```
10  DEF FNA(X) = X + 8
20  DEF FNB(Y) = 2 + Y
30  W = 5
40  C = FNA(W) + FNB(W)
50  PRINT C
60  END
```

Determine the output after making the following change to Line 40 in the above program:

2. 40 C = FNA(3) * FNB(2*W)

3. 40 C = FNB(7*W) - FNA(W ^2)

4. 40 C = 15 + FNA(Y) - FNB(X)

5. 40 C = FNA(1.5) + FNB(W/2)

6. 40 C = FNB (7.25) + FNA(8.25 - W)

7. 40 C = 3*FNB(12 - W) - 2*FNA(W + 8)

8. 40 C = FNA(FNB(FNA(FNB(W - 3))))

9. 40 C = FNB(FNA(FNB(FNA(W + 3))))

10. 40 C = INT(FNA(20)/FNB(W - 2))

11. 40 C = FNA(INT(INT(7*RND(1) + 17)/8))

12. 40 C = INT(SQR(FNA(4*W) + FNB(20)))

---

*From Duplicating Masters -- Experiencing BASIC by Michael Mulcahy, page 12;

Handout #5
Functions and Graphics

NAME _____
DATE _____
PERIOD _____

USER-DEFINED FUNCTIONS
INDIVIDUAL PROBLEMS

You are not limited to the functions provided by BASIC. You may create your own. These programmer-defined functions are restricted to a length of one line each and should be declared at the beginning of the program. The name of the function must start with the letters FN followed by any letter. The argument must be a single numeric data name and the formula should contain that data name:

```
EX  100 DEF FNA(X) = X + 3
200 LET X = 20
300 C = FNA(X)
400 PRINT C
```

Do Two of the Following Programs Using USER-DEFINED Functions.

1A. Write a program to convert Fahrenheit to Celsius and Celsius to Fahrenheit.

OR (for students with background in trigonometry)

1B. Write a program to convert an angle from degree measurement to radian measurement.

2. Write a program to find the radius of a circle, given the area. Modify the problem with a user-defined function to round the radius off to hundredths.

3A. The Ajax Discount Center offers a 20 percent discount for a purchase of $100 or more and a 15 percent discount for a purchase of less than $100. Write a program that uses two one-line functions to determine the final selling price given the original price.*

OR (for students with background in trigonometry)

3B. Write a program to produce two columns of the sines and cosines of complementary angles from 1 to 90. Be sure to employ a user-defined function.

---

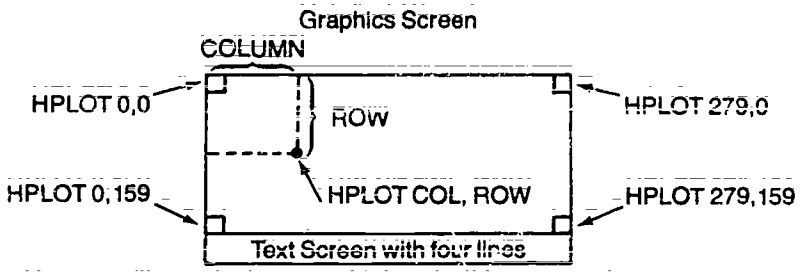
*From Programming in Apple BASIC by John J. Dielsi, Elaine S. Grossman, John P. Tucciarone, page 312; copyright (c) 1984 by CBS College Publishing. New York, New York. Reproduced with permission.

HIGH-RESOLUTION GRAPHICS SUMMARY SHEET *

HCOLORs

Reference Number	HCOLOR	Restrictions
0	black	
1	green	Plotted only if column location is an odd number.
2	purple	Plotted only if column location is an even number.
3	white	Density and color will differ for odd and even column locations, and will be influenced by adjacent colored points.
4	black	
5	orange	Plotted only if column location is an odd number.
6	blue	Plotted only if column location is an even number.
7	white	Same as for HCOLOR = 3.

Note: The HCOLORs can vary greatly, depending on the tuning of your color monitor or color television set.



Home position on text screen obtained by this command sequence:

```
HOME
VTAB 21
HTAB 1
```

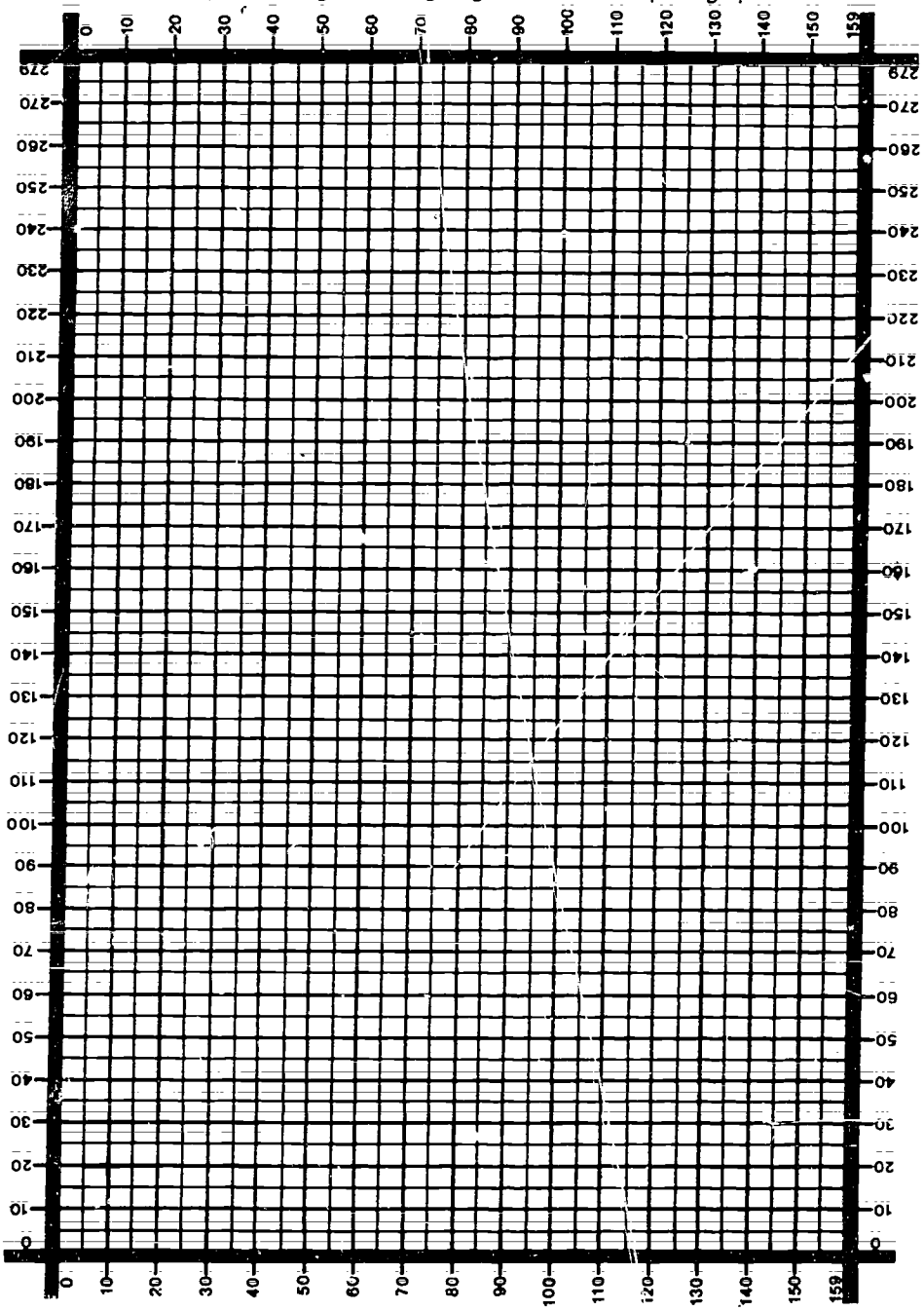
Sample Program to Illustrate Use of Each Graphics Command

Command	Comment
10 HGR	Clears screen (except for bottom four text lines) and enters high-resolution graphics mode.
20 HCOLOR= 1	Establishes green as the high-resolution plot color.
30 HPLOT 15,22	Plots a green point at the intersection of the 15th column and the 22nd row.
40 HOME	Clears text screen and moves the cursor to the upper left corner of the full screen (not the text window).
50 TEXT	Resets the full screen to the text mode.
60 END	

*From Graphics Discoveries -- Book II Jerry Johnson, page 17; copyright (c) 1984 by Creative Publications, Palo Alto, California. Reproduced with permission.

HIGH-RESOLUTION GRAPHICS DESIGN GRID *

Use copies of this grid to design high-resolution computer graphics.



*From Graphics Discoveries -- Book II by Jerry Johnson, page 32; copyright (c) 1984 by Creative Publications, Palo Alto, California. Reproduced with permission.

NAME_____
DATE_____
PERIOD_____

SAMPLE DEMO HI-RES GRAPHICS PROBLEMS*

1. Read this program and predict the computer's output. Then check your prediction by running the program.

```
010 HOME                                    Predicted Output:
020 HGR
030 HCOLOR = 1
040 FOR ROW = 0 TO 159
050 HPLOT 0, ROW
060 HPLOT TO 279, ROW
070 NEXT ROW
080 HCOLOR = 2
090 FOR N = 0 TO 100
100 HPLOT N,N
110 NEXT N
120 END
```

2. Do you like surprises? If so, you'll enjoy running this program. But before you do run it, predict what you think will happen.

```
010 HOME                                    Predicted Output:
020 HGR
030 HCOLOR = 1
040 FOR ROW = 0 TO 159
050 HPLOT 0, ROW
060 HPLOT TO 279, ROW
070 NEXT ROW
080 HCOLOR = 4
090 FOR COL = 0 TO 279 STEP 2
100 FOR ROW = 0 TO 159
110 HPLOT COL, ROW
120 NEXT ROW
130 NEXT COL
140 END
```

What does occur?

Can you explain why the results are so weird?

100

Try making these changes:

```
90  FOR COL = 0 TO 279 STEP 1
```

or

```
90  FOR COL = 0 TO 279 STEP 3
```

or

```
90  FOR COL = 0 TO 279 STEP 10
```

Now do some exploring on your own. See what other visual effects you can create by making changes in the original program. Try new color combinations and different STEP values in line 90.

3. Read this program and predict the computer's output. Then check your prediction by running the program. Explore the effects of changing the program slightly.

```
10  HOME                                    Predicted Output:
20  HGR
30  FOR ROW = 0 TO 159
40  HCOLOR = 5
50  HPLOT 0, ROW TO ROW, 0
60  HCOLOR = 2
70  HPLOT ROW, 159 TO 159, ROW
80  NEXT ROW
90  END
```

What will happen if you add these four lines?

```
54  IF ROW + 160 >= 279 THEN 60          Predicted Output:
55  HPLOT 160, ROW TO ROW + 160, 0
74  IF ROW + 160 >= 279 THEN 80
75  HPLOT ROW + 160, 120 TO 279, ROW
```

What will happen if you change lines 50 and 55 to:

```
50  HPLOT 0, 159 - ROW TO 159 - ROW, 0   Predicted Output:
55  HPLOT 160, 120 - ROW TO 279 - ROW, 0
```

---

*From Graphics Discoveries -- Book II by Jerry Johnson, pages 23-24, 29; copyright (c) 1984 by Creative Publications, Palo Alto, California. Reproduced with permission.

NAME _____
DATE _____
PERIOD _____

SAMPLE DEMO HI-RES GRAPHICS
PROBLEM TO GRAPH POLYNOMIALS

```
010  HOME
020  REM  HI-RES GRAPHICS DEMO
030  REM
040  REM  VARIABLE ASSIGNMENTS
050  REM    W$ = CHARACTER STRING
060  REM    G$ = KEYBOARD RESPONSE
070  REM     A = LOOP VARIABLE
080  REM    SC = SCREEN X-COORDINATE (COLUMN)
090  REM    SR = SCREEN Y-COORDINATE (ROW)
100  REM    OX = SCREEN X-COORD (COLUMN) FOR ORIGIN
110  REM    OY = SCREEN Y-COORD (ROW) FOR ORIGIN
120  REM    SX = SCALAR FOR X-COORD
130  REM    SY = SCALAR FOR Y-COORD
140  REM     N = DEGREE OF THE POLYNOMIAL
150  REM  FN C = POLYNOMIAL OF Nth DEGREE
160  REM     X = X-COORD FOR EQUATION
170  REM     Y = Y-COORD FOR EQUATION
180  REM     P = DIMENSION VARIABLE FOR COEFFICIENT OF POLYNOMIAL
190  REM    R$ = RESPONSE FOR OPTION TO CONTINUE
200  REM
210  REM  TITLE SCREEN
220  INVERSE
230  LET W$ = "RESOLUTION"
240  HTAB (17): VTAB (2): PRINT "H I G H"
250  FOR A = 1 TO 10
260  FLASH
270  HTAB (A + 15): VTAB (A + 3): PRINT  MID$ (W$,A,1)
280  NEXT A
290  INVERSE
300  HTAB (13): VTAB (16): PRINT "G R A P H I C S"
310  VTAB (22)
320  NORMAL
330  PRINT "PRESS ANY KEY TO CONTINUE.": GET G$
340  REM
350  REM  DIMENSION VARIABLE FOR POLYNOMIAL
360  DIM P(50)
370  REM
380  REM  INITIALIZATION OF VARIABLES
390  LET SC = 0
400  LET SR = 0
410  REM
420  REM  INPUT MODULE FOR ORIGIN AND SCALAR
430  TEXT : HOME
440  PRINT "WHAT POINT ON THE SCREEN WOULD YOU LIKE THE ORIGIN?"
```

```
450  PRINT
460  PRINT "ENTER THE COLUMN:";: INPUT OX
470  PRINT "          THE ROW:";: INPUT OY
480  PRINT : PRINT : PRINT
490  PRINT "WHAT IS THE SCALAR VALUE FOR THE COORDINATES?"
500  PRINT
510  PRINT "ENTER THE SCALAR FOR"
520  PRINT
530  PRINT "        THE X-COORDINATE:";: INPUT SX
540  PRINT "        THE Y-COORDINATE:";: INPUT SY
550  REM
560  HOME
570  REM  INPUT MODULE FOR EQUATION YOU WANT GRAPHED
580  PRINT : PRINT
590  PRINT "WHAT DEGREE POLYNOMIAL WOULD YOU LIKE GRAPHED? (BETWEEN 0
     AND 5)";
600  INPUT N
610  REM
620  REM  LOOP TO INPUT COEFFICIENTS OF POLYNOMIALS
630  REM
640  REM  STATEMENT TO INPUT CONSTANT TERM OF POLYNOMIAL
650  IF N = 0 THEN  GOTO 720
660  PRINT "WHAT IS THE COEFFICIENT OF:"
670  FOR A = N TO 1 STEP - 1
680  PRINT
690  PRINT "X TO THE ";A;"th POWER";:INPUT P(A)
700  NEXT A
710  PRINT
720  PRINT "WHAT IS THE CONSTANT TERM";:INPUT P(0)
730  ON N + 1 GOSUB 1220,1260,1300,1340,1380,1420
740  REM
750  REM  DRAWING AXES
760  HGR
770  HCOLOR= 3
780  HPLOT 0,OY TO 279,OY
790  HPLOT OX,0 TO OX,159
800  REM
810  REM  LOOP TO PRINT POLYNOMIAL IN TEXT WINDOW
820  HOME
830  VTAB (21)
840  PRINT "y = ";
850  REM  CHECK FOR CONSTANT POLYNOMIAL
860  IF N = 0 THEN  GOTO 940
870  FOR A = N TO 1 STEP - 1
880  REM  CHECK FOR COEFFICIENTS OF 0 AND 1
890  IF P(A) = 0 THEN  GOTO 930
900  IF P(A) = 1 THEN  GOTO 920
910  PRINT P(A);
920  PRINT "x^";A;" + ";
930  NEXT A
940  PRINT P(0)
950  REM
```

```
0960  REM  PLOTTING POINTS OF POLYNOMIAL
0970  FOR SC = 0 TO 279 STEP 2
0980  LET X = (SC - OX) / SX
0990  LET Y = FN C(X)
1000  LET SR = OY - SY * Y
1010  IF SR<0 OR SR>159 THEN 1030
1020  HPLOT SC,SR
1030  NEXT SC
1040  REM
1050  REM  OPTION TO CONTINUE
1060  PRINT : PRINT
1070  PRINT "DO YOU WANT TO CONTINUE";: INPUT R$
1080  IF R$ = "YES" OR R$ = "Y" THEN  GOTO 390
1090  IF R$ = "NO" OR R$ = "N" THEN 1130
1100  PRINT "INPUT DOES NOT REGISTER!...TRY AGAIN"
1110  GOTO 1070
1120  REM  ENDING SCREEN
1130  TEXT : HOME
1140  VTAB (15) : HTAB (18)
1150  FLASH
1160  PRINT "ALOHA"
1170  NORMAL
1180  REM
1190  END
1200  REM
1210  REM  SUBROUTINES FOR POLYNOMIALS
1220  REM  POLYNOMIAL OF ZERO DEGREE
1230  DEF  FN C(X) = P(0)
1240  RETURN
1250  REM
1260  REM  POLYNOMIAL OF FIRST DEGREE
1270  DEF  FN C(X) = P(1) * X + P(0)
1280  RETURN
1290  REM
1300  REM  POLYNOMIAL OF SECOND DEGREE
1310  DEF  FN C(X) = P(2) * X^ 2 + P(1) * X + P(0)
1320  RETURN
1330  REM
1340  REM  POLYNOMIAL OF THIRD DEGREE
1350  DEF  FN C(X) = P(3) * X^ 3 + P(2) * X^ 2 + P(1) * X + P(0)
1360  RETURN
1370  REM
1380  REM  POLYNOMIAL OF FOURTH DEGREE
1390  DEF  FN C(X) = P(4) * X^ 4 + P(3) * X^ 3 + P(2) * X^ 2 + P(1) * X +
      P(0)
1400  RETURN
1410  REM
1420  REM  POLYNOMIAL OF FIFTH DEGREE
1430  DEF  FN C(X) = P(5) * X^ 5 + P(4) * X^ 4 + P(3) * X^ 3 + P(2) * X^ 2 +
      P(1) * X + P(0)
1440  RETURN
```

NAME _____
DATE _____
PERIOD _____

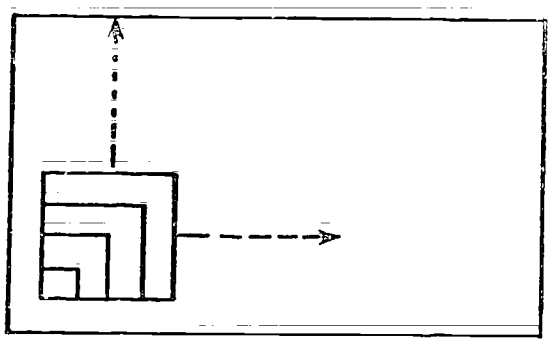
HI-RES GRAPHICS PROBLEMS*

Required Problem.

1. Write a program that will:

   (A) start at an anchor point located at column 5, row 155.
   (B) draw a progression of nested square frames in white (HCOLOR = 3). The sides of each new square frame are 5 "points" longer than the sides of the previous square frame.
   (C) have the anchor point serve as the lower-left corner of each square.
   (D) stop when the top of the graphics screen is reached.

   Hint: Figure out how the locations of the four vertices of the square frame are related to each other.



Optional Problem.

2. Change the program in problem 1 so that it repeats itself continuously and incorporates these changes:

   (A) the square frames are orange.

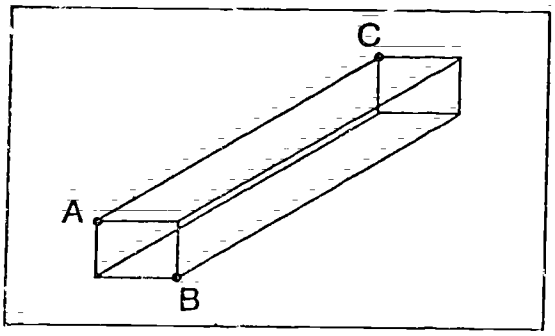
   (B) a new anchor point is selected at random for each new progression of nested squares.

   Note: It is best to limit the row and column values of the anchor points to multiples of 5.

Required Problem.

3. A rectangular parallelepiped is a solid figure with six rectangular faces. Write a program that will draw the outline of a rectangular parallelepiped in white (HCOLOR = 3). The program should allow you to input the row and column values of three key vertices A, B, and C.

   Hint: Analyze the drawing below and figure out the locations of the remaining five vertices. Remember that once the location of vertex A is chosen, the available locations of vertices B and C will be restricted somewhat.



Vertex	Column	Row
A	C1	R1
B	C2	R2
C	C3	R3

Optional Problem.

4. Change the program in problem 3 so that the locations of vertices A, B, and C are chosen randomly. After one rectangular parallelepiped is drawn, you should be able to press the RETURN key in order to clear the screen and draw a new parallelepiped.

---

*From Graphics Discoveries -- Book II by Jerry Johnson, pages 35, 37; copyright (c) 1984 by Creative Publications, Palo Alto, California. Reproduced with permission.

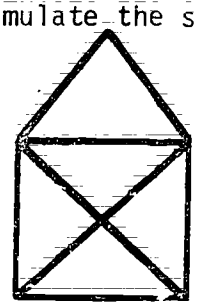
NAME_____
DATE_____
PERIOD_____

HI-RES GRAPHICS INDIVIDUAL PROGRAMS

1 Complete the drawing of a chessboard by shading in every other square in white using high-resolution graphics.

For extra credit you might draw a few of the chess pieces under your picture of the board!

2. This shape may be drawn without lifting your pencil from the paper or retracing any lines. Simulate the solution to this exercise using high resolution graphics.



3. Write a program that produces a slide show of optical illusions. Some good illusions to include are:

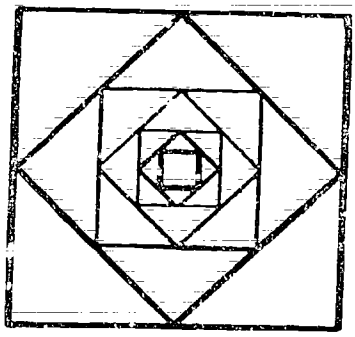
a. the Zollner illusion
b. the Necker Cube illusion
c. the Poggendorf illusion
d. the Ponzo or railway lines illusion
e. the Hering illusion and its converse
f. Penrose's Impossible Triangle

These and other puzzling illusions can be found in the following references:

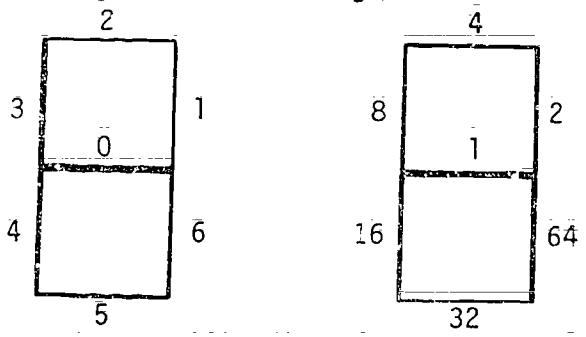
Gilliam, B. "Geometrical Illusions." Scientific American, (January, 1980), pp. 102-111.
Gregory, R. "Visual Illusion." Scientific American, (November, 1968), pp. 66-76.
Gregory, R. The Intelligent Eye. New York: McGraw-Hill, 1970.
Lanners, E., ed. Illusions. New York: Holt, Rinehart, and Winston, 1977.
Levine, S. "Optical Illusions." Grade Teacher. (October, 1970), pp. 74-77.*

---

4. Write a program to produce a design similar to this. Have the program draw the design several times using different colors each time. You are to use high-resolution graphics



5. A seven segment display is used to represent a number on a digital clock. The seven segments are arranged as shown below:



When certain segments are lit, they form a numeral. For example, if a figure five is to be represented, then segments 0, 2, 3, 5, and 6 would be lit. If each segment is assigned a value of a power of two as shown above, it is possible to encode the form of a numeral as the sum of the values of the segments that must be lit to represent that numeral. For example, the encoded information of a figure five would be the sum of the segments 0, 2, 3, 5 and 6 or 1+4+8+32+64 = 109.

To decode a number like 109, successively divide it by two. If the result of the first division is not an integer, then the segment numbered 0 should be lit. Again take the INT of the result. Continue this process of dividing by two and checking the result seven times (once for each segment).

Write a program to have the computer count slowly from 0 to 9 using the seven segment display described above. For extra credit, modify the program to make it count to 99.*

---

*From A Guide to Programming in Applesoft BASIC by Bruce Presley, page 6.21; copyright (c) 1984 by Lawrenceville Press, Lawrenceville, New Jersey. Reproduced with permission.

SAMPLE ACTIVITY CLUSTER #5
FILE HANDLING AND TERM PROJECT

Topics:

1. Sequential Files
2. Random or Direct Files
3. Term Project

Classroom Management:

The work on files can be done by pairs and teams of students, but the term project should be done individually.

Materials:

Demonstration microcomputer and large-screen monitor;
Videotape player and the videotape, "File Structures";
Reading material on Text Files, such as:
Apple Text Files by David Miller, chapters 1-3;
Data File Programming in BASIC -- A Self-Teaching Guide by LeRoy Finkel and Jerald R. Brown, chapters 4, 7-8;
Handouts.

Time for Activities:

Approximately one to four weeks for file handling could be provided, depending on the depth to which the teacher plans taking his or her students. Four to six weeks could be allowed for the term project; specific days should be set for work on the projects.

Teacher Preparation and Procedures:

Preview the videotape and prepare the needed handouts. Gather an assortment of topic ideas for the term projects. All sample materials for these activities are located in the Sample Assignments and Materials for Activity Cluster #5.

1. Introduce text files and file handling via a lecture and reading assignment. Discuss related questions.

2. To familiarize students with the actual process of creating and using data files, have pairs of students type in a couple, simple, sample programs, using a sequential file. Distribute Handout #1 - Sample Programs Using Sequential Files. Remind students of the purpose of the CHR$ (4) string function when handling files in Applesoft BASIC.

3. Show the videotape, "File Structures," and have students answer questions from Handout #2 - Questions from the Videotape, "File Structures." Take time to discuss the answers.

109

4. Assign pairs of students one of two programs that create and read back sequential data files, found in Handout #3 - Creating and Reading Back Sequential Data Files. Require a flowchart and good documentation. The solutions, adjusted to Applesoft BASIC, may be removed from the handout or given to students after they have written their own program. As various versions of BASIC open, write and close files differently, appropriate adjustments must be made for your particular system. For students that may want to include a menu in their program, a sample program and hierarchy chart are provided as a guide for this arrangement in Handout #4 - Creating and Maintaining Sequential Files Using Menus. The program is written for TRS-80, Model 4 and would need some adjustments for other systems.

5. After exposing students to the differences between random access files and sequential files, provide a series of three interrelated programming problems that create and read random access data files. Refer to Handout #5 - Random Access Data Files. The sample listings may be removed from the handout or given to students after they have written their own program. Assign all pairs of students to one of the three problems according to their level of skills. For the class to gain experience in a real programming environment, organize teams with three pairs of students--each pair responsible for one of the three programs but the entire team responsible for the overall system. Emphasize the need for cooperation within pairs and among team members in planning, designing, coding and testing their programs. Have teams appoint one member as their lead programmer--not to do all the work but to coordinate the three pairs of members within the team and assist where needed.

6. As an option, discuss the advantages of an indexed-sequential file over a regular sequential file or random access file. Assign Handout #6 - Creating an Indexed-Sequential File. A listing of a possible solution to the problem for TRS-80, Model 4 is included.

7. A term project could be assigned any time during the second semester, after students have become more adept at BASIC programming skills. Handout #7 - Term Project could be distributed when appropriate.

Sample Assignments and Materials for Activity Cluster #5:

The following pages in this activity cluster provide handouts that could be used in this course.

NAME_____
DATE_____
PERIOD_____

SAMPLE PROGRAMS USING SEQUENTIAL FILES

Example 1

The program below is divided into three sections. The first section allows the user to input the name of the file to be created. This name is stored in F$. Thereafter, rather than typing this name with each file command, F$ is used. See lines 2060, 2070, 2080, 2090, and 2140.

```
0080 DIM A$(100)
0090 D$ = CHR$(4)
0100 REM  ENTER FILE NAME
0110 REM
0120 HOME
0130 PRINT "ENTER NAME OF FILE TO BE CREATED:"
0140 VTAB 4:  HTAB 15:  INPUT F$
0150 PRINT ************************************************
1000 REM INPUT DATA INTO AN ARRAY TO BE RECORDED ON FILE
1010 REM
1020 C = 0
1025 VTAB 6
1030 PRINT "RECORD ";C + 1;
1040 INPUT " ";A$(C + 1)
1050 IF A$(C + 1) = " " THEN 2020
1060 C = C + 1
1070 GOTO 1030
1080 REM  ************************************************
2000 REM  TRANSFER ARRAY DATA TO FILE
2010 REM
2020 HOME
2030 VTAB 13
2040 PRINT C;" RECORDS HAVE BEEN CREATED"
2050 VTAB 15:  PRINT "FOR FILE ";F$
2060 PRINT D$;"OPEN ";F$
2070 PRINT D$;"DELETE ";F$
2080 PRINT D$;"OPEN ";F$
2090 PRINT D$;"WRITE ";F$
2100 PRINT C
2110 FOR N = 1 TO C
2120 PRINT A$(N)
2130 NEXT N
2140 PRINT D$;"CLOSE ";F$
```

The second section of the program stores the information for the file and A$. Line 80 allows for a maximum of 100 such records. If a larger array size were required, only line 80 would have to be modified. The end of a

record is indicated by the return key following each INPUT. The counter C in line 1060 keeps track of the number of records to be written to the file, and line 1050 terminates this section. When all the data have been entered, the user will just hit the RETURN key and, as a result, the null string stored in the last A$(C+1) will indicate that no more data are to be entered. Note that since the counter in line 1060 is not incremented in this case, the null string will not be included as part of the text file.

The third section of the program takes the data accumulated in the second section and stores it on the diskette. The first data item recorded on the diskette is C, the number of records in the file. Remember, this data item is stored as record 0 in this sequential file. The remaining data items are stored by using the loop in lines 2110 to 2130. Since the PRINT statement in this loop follows a WRITE file command, it outputs data to the file designated F$.

This example illustrates one method of creating a sequential data file. In order to access the information stored on the file, the programmer must know how the file was created.

The example below indicates how one can access a file created in manner similar to the method used in Example 1. Note that the first record read is A, the variable representing the total number of records in the file.

Example 2

```
0080 DIM B$(100)
0090 D$ = CHR$(4)
0100 REM  ENTER FILE NAME
0110 REM
0120 HOME
0130 PRINT "ENTER NAME OF FILE TO BE READ:"
0140 VTAB 4: HTAB 15:  INPUT F$
0150 PRINT
0160 REM  ************************************************
1000 REM  READ DATA FROM FILE
1010 REM
1020 PRINT D$;"OPEN ";F$
1030 PRINT D$;"READ ";F$
1040 INPUT A
1050 PRINT "THERE ARE ";A;" RECORDS IN FILE ";
1060 INVERSE
1070 PRINT F$
1080 NORMAL
1090 PRINT
1100 FOR C = 1 TO A
1110 INPUT B$(C)
1120 PRINT "RECORD ";C;": ";B$(C)
1130 NEXT C
1140 PRINT D$;"CLOSE ";F$
```

Handout #2
File Handling and Project

NAME_____
DATE_____
PERIOD_____

QUESTIONS FOR THE VIDEOTAPE,
"File Structures"

1. How does data storage on disk differ from data storage on tape?

2. How are sequential files different from random files?

3. For what types of applications are sequential files best suited?

4. In Applesoft BASIC, what purpose does the string function CHR$(4) serve?

5. Explain the following file commands:

OPEN	READ
CLOSE	WRITE
DELETE	FILE

6. Besides the name of the files, what additional information must be provided for accessing random files?

7. Of what value is the command MON? What subcommands can be entered under MON?

NAME_____
DATE_____
PERIOD_____

CREATING AND READING BACK SEQUENTIAL DATA FILES*

1A Write a program to create a data file called GROCERY that stores your grocery shopping list. Include the description or name of each grocery item (maximum of twenty characters) and a numeric value telling the quantity of that item to buy. Store at least six datasets or entries in the file. Remark statements should identify the program and variables, such as:

```
100 REM     GROCERIES
110 REM
120 REM
130 REM     VARIABLES USED
140 REM       I$ = ITEM DESCRIPTION
150 REM       Q  = QUANTITY TO ORDER
160 REM
180 REM     FILES USED
200 REM       F$ = USER ENTERED INPUT FILE
```

1B Write a companion program to display the contents of GROCERY.

Possible Solutions to 1A and 1B
Using Applesoft BASIC

1A GROCERY SHOPPING LIST

```
110  REM PROB 1A SOLUTION (GROCERY LIST)
120  REM
130  REM INTRODUCTORY MODULE
140  REM  VARIABLES USED
150  REM     I$ = ITEM DESCRIPTION
160  REM     Q  = QUANTITY TO ORDER
170  REM
180  REM  FILES USED
190  REM     F$ = USER ENTERED INPUT FILE
200  REM
220  D$ = CHR$(4)
230  INPUT "ENTER NAME OF INPUT FILE"; F$
240  PRINT D$;"OPEN ";F$
250  REM
280  REM  DATA ENTRY ROUTINE
290  REM
300  PRINT "ENTER 'STOP' WHEN FINISHED"
310  PRINT
320  INPUT "ENTER ITEM DESCRIPTION:"; I$
330  IF I$ = "STOP" THEN 480
340  IF LEN(I$) = 0 THEN PRINT "PLEASE ENTER A DESCRIPTION OR 'STOP'": GOTO
     320
350  IF LEN(I$) >= 20 THEN PRINT "PLEASE LIMIT DESCRIPTION TO 20 CHARS, AND
     REENTER": GOTO 320
360  INPUT "ENTER QUANTITY:"; Q
370  IF Q >= 1 AND Q <= 10 THEN 440
380  PRINT "YOU ENTERED A QUANTITY OF "; Q
390  INPUT "IS THAT WHAT YOU WANTED?"; R$
400  IF LEFT$(R$,1) = "N" THEN 360
410  REM
420  REM   WRITE TO FILE ROUTINE
430  REM
440  PRINT D$;"WRITE ";F$
445  PRINT I$
450  PRINT Q
455  GOTO 320
460  REM
470  REM   CLOSE FILE ROUTINE
480  PRINT D$;"CLOSE ";F$
485  REM
490  PRINT "FILES CLOSED."
500  END
```

1B GROCERY SHOPPING LIST DISPLAY

```
100  REM  PROB 1B SOLUTION (GROCERY LIST FILE APPLICATION)
110  REM
120  REM  VARIABLES USED
130  REM     I$ = ITEM DESCRIPTION
140  REM     Q  = QUANTITY TO ORDER
150  REM
160  REM  FILES USED
170  REM     F$ = USER ENTERED INPUT FILE
180  REM
190  REM  FILES INITIALIZATION ROUTINE
195  D$ = CHR$(4)
200  INPUT "ENTER NAME OF INPUT FILE:"; F$
210  PRINT D$;"OPEN ";F$
220  REM
230  REM  READ/PRINT FILE ROUTINE
240  REM
250  PRINT "ITEM", "QUANTITY": PRINT
260  PRINT D$;"READ ";F$
265  INPUT I$
270  INPUT Q
275  IF I$ = "STOP" THEN 320
280  PRINT I$,Q
290  GOTO 265
300  REM
310  REM  CLOSE FILE ROUTINE
320  PRINT D$;"CLOSE ";F$
330  END
```

2A Write one program and use it to create three different data files called LETTER1, LETTER2, and LETTER3. Each file should contain the text of a form letter with at least three lines of text per letter. Each line of text in the letters is to be entered and stored as one dataset or entry.

```
100 REM LETTERS
110 REM
120 REM VARIABLES USED
130 REM  R$ = TEXT LINE
140 REM  F$ = FILE NAME VARIABLE
150 REM
160 REM FILES USED
170 REM  LETTER (PLUS F$ WHICH IS USER SELECTED)
180 REM
```

2B Write a companion program to display the data file above selected by the user.

117

Possible Solutions to 2A and 2B
Using Applesoft BASIC

2A LETTERS

```
100 REM  PROB 2A SOLUTION (LETTERS)
110 REM
120 REM  VARIABLES USED
130 REM   R$ = TEXT LINE
140 REM   F$ = FILE NAME VARIABLE
150 REM
160 REM  FILES USED
170 REM     LETTER (PLUS F$ WHICH IS USER SELECTED)
180 REM
190 REM   INITIALIZE ROUTINE
200 REM
210 CLEAR
215 D$ = CHR$(4)
220 INPUT "ENTER FILE NUMBER:"; F$
230 LET F$ = "LETTER" + F$
240 PRINT D$;"OPEN ";F$
250 REM
260 PRINT "ENTER TEXT LINE OR 'STOP'"
270 INPUT R$
280 IF R$ = "STOP" THEN 330
290 PRINT D$;"WRITE ";F$
295 PRINT R$
300 GOTO 260
310 REM
320 REM  CLOSE FILE ROUTINE
330 PRINT D$;"CLOSE ";F$
340 PRINT "FILE ";F$;" CLOSED."
350 END
```

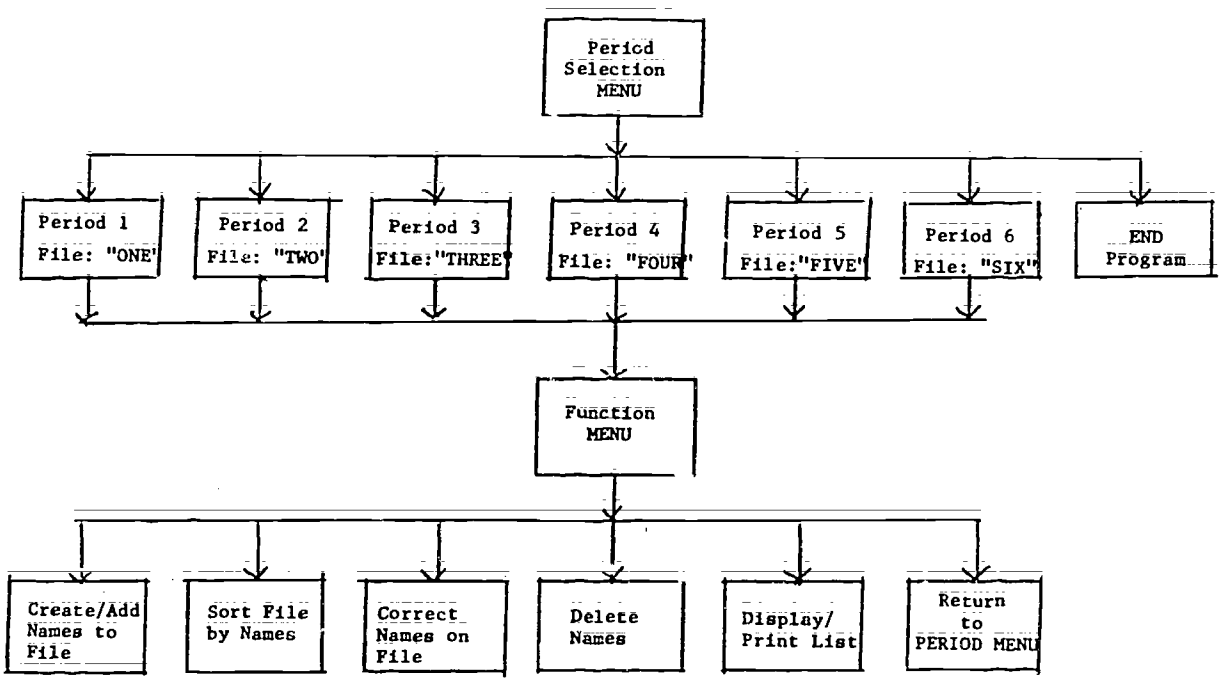
2B LETTERS DISPLAY

```
100  REM  PROB 2B SOLUTION (LETTERS FILE APPLICATION)
110  REM
120  REM  VARIABLES USED
130  REM   R$ = TEXT LINE
140  REM   F$ = FILE NAME VARIABLE
150  REM
160  REM  FILES USED
170  REM     LETTER (PLUS F$ WHICH IS USER SELECTED)
180  REM
190  REM  INITIALIZE ROUTINE
195  REM
200  CLEAR
205  D$ = CHR$(4)
210  INPUT "ENTER FILE NUMBER:"; F$
220  LET F$ = "LETTER" + F$
230  PRINT D$;"OPEN ";F$
240  REM
250  REM   READ/PRINT FILE CONTENTS ROUTINE
255  REM
260  PRINT D$;"READ ";F$
265  INPUT R$
270  IF R$ = "STOP" THEN 320
280  PRINT R$
290  GOTO 265
300  REM
310  REM   CLOSE FILE ROUTINE
320  PRINT D$;"CLOSE ";F$
330  PRINT "FILE ";F$;" CLOSED."
340  END
```

*Adapted from Data File Programming in BASIC by LeRoy Finkel and Jerald R. Brown, pages 113-114, 124, 126, 128-129 and 133-134; copyright (c) 1981 by John Wiley and Sons, Inc., New York, New York. Reproduced with permission.

NAME_____
DATE_____
PERIOD_____

CREATING AND MAINTAINING SEQUENTIAL FILES USING MENUS
Hierarchy Chart
Class Files



CLASS FILES PROGRAM

```
0100 REM  THIS PROGRAM CREATES AND MAINTAINS SEQUENTIAL CLASS FILES BY
0110 REM  PERIODS USING TWO MENUS:  PERIOD MENU AND FILE MAINTENANCE MENU
0120 REM  WRITTEN FOR A TRS-80 MODEL 4 COMPUTER W/ WHILE-WEND & SWAP
0130 '
0140 REM***VARIABLE NAMES USED***
0150 '    F$=NAME OF SEQUENTIAL DATA FILE
0160 '    STU.NAME$=STUDENT'S NAME
0170 '    S.NAME$=NAME TO BE CORRECTED
0180 '    S.N$=CORRECT NAME
0190 '    SN$=NAME TO BE DELETED
0200 '    GRADE$=GRADE LEVEL OF STUDENT
0210 '    GR$=GRADE LEVEL CORRECTION
0220 '    S/N/S1/S2=SUBSCRIPTS and COUNTERS
0230 '
0240 '
0250 DIM STU.NAME$(40), GRADE$(40)
0260 '
0500 REM***PERIOD SELECTION MENU***
0510 CLS
0520 PRINT "     PERIOD MENU": PRINT
0530 PRINT "CODE     FUNCTION"
0540 PRINT "  1   UPDATE PERIOD 1 CLASS LIST"
0550 PRINT "  2                 2 CLASS LIST"
0560 PRINT "  3                 3 CLASS LIST"
0570 PRINT "  4                 4 CLASS LIST"
0580 PRINT "  5                 5 CLASS LIST"
0590 PRINT "  6                 6 CLASS LIST"
0600 PRINT "  7 END PROGRAM"
0610 PRINT: PRINT "WHICH PERIOD DO YOU WISH TO CREATE/UPDATE? ";
0620 INPUT "ENTER A NUMBER 1 THROUGH 7: "; PERIOD
0630 IF PERIOD>=1 AND PERIOD<=7 THEN 0680
0640 PRINT
0650 PRINT "   PERIOD "; PERIOD " IS INVALID"
0660 INPUT "   PLEASE RE-ENTER 1, 2, 3, 4, 5, 6 OR 7: "; PERIOD: GOTO 630
0680 IF PERIOD = 1 THEN F$="ONE": GOTO 0750
0690 IF PERIOD = 2 THEN F$="TWO": GOTO 0750
0700 IF PERIOD = 3 THEN F$="THREE": GOTO 0750
0710 IF PERIOD = 4 THEN F$="FOUR": GOTO 0750
0720 IF PERIOD = 5 THEN F$="FIVE": GOTO 0750
0730 IF PERIOD = 6 THEN F$="SIX": GOTO 0750
0740 IF PERIOD = 7 THEN 0760
0750 GOTO 1000
0760 PRINT: PRINT "END OF CLASS ROSTER PROGRAM"
0770 END
1000 REM***MENU FOR FILE CREATION AND MAINTENANCE***
1010 CLS
1020 PRINT "    FILE CREATION/MAINTENANCE MENU": PRINT
1030 PRINT "CODE     FUNCTION"
```

```
1040 PRINT "  1   CREATE AND ADD NAMES TO CLASS LIST"
1050 PRINT "  2   SORT NAMES IN CLASS LIST"
1060 PRINT "  3   CHANGE/CORRECT NAME(S) ON CLASS LIST"
1070 PRINT "  4   DELETE NAME(S) FROM CLASS LIST"
1080 PRINT "  5   PRINT/DISPLAY NAMES ON CLASS LIST"
1090 PRINT "  6   RETURN TO MAIN MENU"
1100 PRINT: PRINT "ENTER A NUMBER 1 THROUGH 6: "
1110 C$=INPUT$(1)
1115 '  MENU SELECTION          (1)   (2)   (3)   (4)   (5)   (6)
1120 ON INSTR("123456", C$) GOTO 2000, 3000, 4000, 5000, 6000, 500
1130 SOUND 2, 0: PRINT "INVALID NUMBER.  PLEASE RE-ENTER 1, 2, 3, 4, 5 OR 6"
     ; : GOTO 1110
1140 '
2000 REM***CREATE AND ADD NAMES TO CLASS LIST***
2010 PRINT "CREATING/UPDATING CLASS LIST FOR PERIOD: "; F$
2030 OPEN "E", 1, F$
2035 CONTINUE$="Y"
2040 WHILE CONTINUE$="Y"
2050    CLS: PRINT
2060    LINE INPUT "ENTER STUDENT'S NAME "; STU.NAME$: PRINT
2080    LINE INPUT "ENTER STUDENT'S GRADE LEVEL "; GRADE$
2090    WRITE #1, STU.NAME$, GRADE$
2100    PRINT: INPUT "MORE NAMES TO ENTER (Y/N) "; CONTINUE$
2110 WEND
2120 CLOSE #1
2130 GOTO 1000
2140 '
3000 REM***SORT CLASS LIST***
3005 ' LOAD CLASS FILE INTO TABLE
3010 OPEN "I", 1, F$
3020 WHILE NOT EOF(1)
3030    LET S=S+1
3040    INPUT #1, STU NAME$(S), GRADE$(S)
3050 WEND
3060 CLOSE #1
3065 CLS: PRINT TAB(10) "SORTING"
3070 FOR S1= 1 TO S-1
3080    FOR S2=S1+1 TO S
3090       IF STU.NAME$(S2)  =STU.NAME$(S1) THEN GOTO 3120
3100       SWAP STU.NAME$(S1), STU.NAME$(S2)
3110       SWAP GRADE$(S1), GRADE$(S2)
3120    NEXT S2
3130 NEXT S1
3135 CLS: PRINT TAB(10) "SORT COMPLETED"
3140 '
3150 REM***REWRITE SORTED CLASS LIST TO FILE***
3160 OPEN "O", 1, F$
3170 FOR N= 1 TO S
3180    WRITE #1, STU.NAME$(N), GRADE$(N)
```

```
3190 NEXT N
3200 CLOSE #1
3210 GOTO 1000
3220 '
4000 REM***CHANGE/CORRECT NAMES ON CLASS LIST***
4005 '  LOAD CLASS FILE INTO TABLE
4010 OPEN "I", 1, F$
4015 S=0
4020 WHILE NOT EOF(1)
4030    LET S=S+1
4040    INPUT #1, STU.NAME$(S), GRADE$(S)
4050 WEND
4060 CLOSE #1
4070 '
4080 CLS
4085 N=0
4090 LINE INPUT "ENTER NAME OF STUDENT YOU WISH TO CHANGE "; S.NAME$
4095 NAME.FOUND$="N"
4100 WHILE NAME.FOUND$<>"Y"
4105    ON ERROR GOTO 5500
4110    LET N=N+1
4115    ON ERROR GOTO 5500
4120    IF S.NAME$<>STU.NAME$(N) THEN 4240
4130    LET NAME.FOUND$="Y"
4150    PRINT "THE STUDENT'S NAME IS "; STU.NAME$(N)
4160    PRINT "GRADE IS "; GRADE$(N)
4170    INPUT "PRESS <ENTER> TO CONTINUE"; R$
4180    PRINT "IF YOU ARE CHANGING NAME AND/OR GRADE, TYPE IN THE CORRECT
        DATA"
4190    PRINT "OTHERWISE, HIT <ENTER> KEY": PRINT
4200    LINE INPUT S.N$
4210    LINE INPUT GR$
4220    IF S.N$="" THEN STU.NAME$(N)=STU.NAME$(N)
           ELSE STU.NAME$(N)=S.N$
4230    IF GR$="" THEN GRADE$(N)=GRADE$(N)
           ELSE GRADE$(N)=GR$
4240 WEND
4270 '
4280 REM***REWRITE FILE TO DISK***
4290 OPEN "O", 1, F$
4300 FOR N= 1 TO S
4310    WRITE #1, STU.NAME$(N), GRADE$(N)
4320 NEXT N
4330 CLOSE #1
4340 GOTO 1000
4350 '
5000 REM***DELETE NAMES FROM CLASS LIST"
5005 '  LOAD CLASS FILE INTO TABLE
5010 OPEN "I", 1, F$
```

```
5020 LET S=0
5030 WHILE NOT EOF(1)
5040    LET S=S+1
5050    INPUT #1, STU.NAME$(S), GRADE$(S)
5060 WEND
5070 CLOSE #1
5080 '
5090 CLS
5100 LINE INPUT "ENTER STUDEN. NAME YOU WISH TO DELETE "; SN$
5100 N=0
5115 FOUND$="N"
5120 WHILE FOUND$<>"Y"
5125 '  ON ERROR GOTO 5500
5130    N=N+1
5140    IF SN$<>STU.NAME$(N) THEN GOTO 5170
5150    FOUND$="Y"
5160    STU.NAME$(N)="ZZZ"
5170 WEND
5200 '
5210 REM***REWRITE CLASS LIST TO FILE***
5220 OPEN "O", 1, F$
5230 FOR N=1 TO S
5250    WRITE #1, STU.NAME$(N), GRADE$(N)
5260 NEXT N
5270 CLOSE #1
5280 GOTO 1000
5290 '
5500 PRINT "NAME NOT FOUND": FOR C=1 TO 1000: NEXT C:  RESUME 1000
6000 REM***PRINT/DISPLAY CLASS LIST***
6010 A$="   ##.    \                    \     \\"
6020 CLS
6030 PRINT "           DISPLAY CLASS LIST"
6040 PRINT: INPUT "DO YOU WANT A HARD COPY (Y/N)"; R$
6050 IF R$="Y" THEN SYSTEM "LINK *DO *PR"
6060 PRINT TAB(15) "CLASS LIST": PRINT
6070 PRINT TAB(15) "PERIOD "; F$ : PRINT
6080 OPEN "I", 1, F$
6090 PRINT "           NAME                        GRADE": PRINT
6100 NO=1
6110 WHILE NOT EOF(1)
6120    INPUT #1, STU.NAME$, GRADE$
6130    IF STU.NAME$="ZZZ" THEN GOTO 6160
6140    PRINT USING A$; NO; STU.NAME$, GRADE$
6150    NO=NO+1
6160 WEND
6165 IF R$="Y" THEN SYSTEM "RESET *DO *PR"
6170 CLOSE #1
6180 GOTO 1000
6190 '
```

NAME_____
DATE_____
PERIOD_____

RANDOM ACCESS DATA FILES*

1. Write a program to create a random access data file, named PRODUCT, that contains the inventory of products carried by an imaginary business. Each random access record contains the following data for one item of inventory in the order shown below. Numbers in parentheses indicate maximum string lengths.

P$ = product number (4)
I$ = description of inventory item (20)
S$ = supplier (20)
L = reorder point (how low the stock of item can be before reordering)
Y = reorder quantity
Q = quantity available (currently in stock)
C = cost (from supplier)
U = unit selling price (what the item is sold for)

Here is the introductory module:

```
100 REM PROB 1 SOLUTION
110 REM
120 REM VARIABLES USED
130 REM   P$ = PROD. NO. (4)
140 REM   I$ = ITEM DESCRIPTION(20)
150 REM   S$ = SUPPLIER (20)
160 REM   L  = REORDER POINT
170 REM   Y  = REORDER QUANTITY
180 REM   Q  = QUANTITY
190 REM   C  = COST
200 REM   U  = UNIT SELLING PRICE
210 REM
220 REM FILES USED
230 REM   PRODUCT (RANDOM ACCESS FILE)
240 REM
```

Using the program, create a random access file. Make up your own data for 20 records (inventory items) and enter them into the file. This file will be needed in problems #2 and #3 for other teams to use.

A flowchart and good documentation are required.

Possible Solution to Problem 1
Using Applesoft BASIC

```
100  REM  PROB 1 SOLUTION
110  REM
120  REM  VARIABLES USED
130  REM    P$ = PROD. NO. (4)
140  REM    I$ = ITEM DESCRIPTION(20)
150  REM    S$ = SUPPLIER(20)
160  REM    L  = REORDER POINT(4)
170  REM    Y  = REORDER QUANTITY(4)
180  REM    Q  = QUANTITY IN STOCK(4)
190  REM    C  = WHOLESALE COST(4)
200  REM    U  = UNIT SELLING PRICE(4)
210  REM    N  = RECORD NUMBER DESIRED
220  REM    I  = INDEX FOR ARRAY
230  REM
235  REM  FILE USED
240  REM    PRODUCT (RANDOM ACCESS FILE)
250  REM  INITIALIZE ROUTINE
260  CLEAR
265  D$ = CHR$(4)
270  PRINT D$;"OPEN PRODUCT,L64"
280  INPUT "HOW MANY RECORDS DO YOU WANT?"; N
285  DIM P$(N),I$(N),S$(N),L(N),Y(N),Q(N),C(N),U(N)
290  REM
300  REM  DATA ENTRY MODULE - DATA ENTRY TESTS OMITTED
305  FOR I = 1 TO N
310  INPUT "ENTER PRODUCT NUMBER (4 DIGITS):"; P$(I)
320  REM*** DATA ENTRY TESTS GO HERE
330  INPUT "ENTER ITEM DESCRIPTION (20 CHAR. MAX.)"; I$(I)
340  REM*** DATA ENTRY TESTS GO HERE
350  INPUT "ENTER NAME OF SUPPLIER (20 CHAR. MAX.)"; S$(I)
360  REM *** DATA ENTRY TESTS GO HERE
370  INPUT "REORDER POINT"; L(I)
380  REM *** DATA ENTRY TESTS GO HERE
390  INPUT "REORDER QUANTITY"; Y(I)
400  REM *** DATA ENTRY TESTS GO HERE
410  INPUT "QUANTITY NOW IN STOCK"; Q(I)
420  REM *** DATA ENTRY TESTS GO HERE
430  INPUT "WHOLESALE COST"; C(I)
440  REM *** DATA ENTRY TESTS GO HERE
450  INPUT "UNIT SELLING PRICE"; U(I)
460  REM *** DATA ENTRY TESTS GO HERE
470  REM *** DATA ENTRY CHECKS DELETED TO SHOW PROGRAM STRUCTURE
480  NEXT I
485  REM
490  FOR I = 1 TO N
495  PRINT D$;"WRITE PRODUCT,R";I
```

```
500  PRINT P$(I)
510  PRINT I$(I)
520  PRINT S$(I)
525  PRINT L(I)
530  PRINT Y(I)
540  PRINT Q(I)
550  PRINT C(I)
560  PRINT U(I)
570  NEXT I
630  REM
640  REM  CLOSE FILE ROUTINE
650  PRINT D$;"CLOSE PRODUCT"
660  PRINT "FILE CLOSED."
670  END
```

*Adapted from Data File Programming in BASIC by LeRoy Finkel and Jerald R. Brown, pages 257 and 261; copyright (c) 1981 by John Wiley and Sons, Inc., New York, New York. Reproduced with permission.

127

NAME_____
DATE_____
PERIOD_____

RANDOM ACCESS DATA FILES*

2. Write two modules for this program, one to create and another to read/display a sequential file named POINT, which has three data items per dataset. The first data item is the product number from each dataset in PRODUCT, the second data item is the value for the quantity in stock, and the third data item is a value that corresponds to the record number in the PRODUCT file where that product number is stored. There should be exactly as many datasets in the sequential file POINT as there are records in the random access file PRODUCT. Your file creating module must input an entire dataset from the random access file named PRODUCT, while only the first and sixth data items and the record number are output to the sequential file named POINT. This file will be needed in problem #3 for use by another team of students. A flowchart and good documentation are required.

Possible Solution to Problem 2
Using Applesoft BASIC

```
100  REM  PROB 2 SOLUTION
110  REM  CREATE A POINTER FILE NAMED 'POINT' FOR RANDOM ACCESS FILE
     'PRODUCT'
120  REM
130  REM  VARIABLES USED
140  REM    P$ = PROD. NO. (5)
150  REM    I$ = ITEM DESCRIPTION (20)
160  REM    S$ = NAME OF SUPPLIER (20)
170  REM    L  = REORDER POINT (4)
180  REM    Y  = REORDER QUANTITY (4)
190  REM    Q  = QUANTITY IN STOCK (4)
200  REM    C  = WHOLESALE COST (4)
210  REM    U  = UNIT SELLING PRICE (4)
220  REM    R  = RECORD NUMBER
230  REM    N  = NUMBER OF RECORDS TO READ
240  REM
250  REM  FILES USED
260  REM    RANDOM ACCESS FILE NAME:  PRODUCT
270  REM    DATASET FORMAT:  P$,I$,S$,L,Y,Q,C,U
280  REM    SEQUENTIAL FILE NAME:  POINT
290  REM    DATASET FORMAT:  P$,Q,R
300  REM
310  REM  INITIALIZE ROUTINE
315  CLEAR
320  D$ = CHR$(4)
330  PRINT D$;"OPEN PRODUCT,L64"
340  PRINT D$;"OPEN POINT"
350  INPUT "HOW MANY RECORDS DO YOU WANT TO READ?";N
355  REM
360  REM  READ 'PRODUCT' AND WRITE 'POINT'
370  FOR R = 1 TO N
380  PRINT D$;"READ PRODUCT,R";R
390  INPUT P$ : INPUT I$
400  INPUT S$ : INPUT L
410  INPUT Y  : INPUT Q
420  INPUT C  : INPUT U
430  PRINT D$;"WRITE POINT"
440  PRINT P$,Q,R
450  NEXT R
455  REM
460  REM  CLOSE FILES ROUTINE
470  PRINT D$;"CLOSE PRODUCT"
490  PRINT D$;"CLOSE POINT"
495  REM
500  REM  READ/DISPLAY 'POINT'
510  PRINT D$;"OPEN POINT"
```

```
520  PRINT D$;"READ POINT"
530  FOR R = 1 TO N
540  INPUT P$ : INPUT Q : INPUT R
550  PRINT "PROD. NO. ";P$;" QUANTITY IN STOCK ";Q;" RECORD NO. ";R
560  NEXT R
565  REM
570  REM CLOSE FILE ROUTINE
580  PRINT D$;"CLOSE POINT"
590  PRINT:  PRINT "ALL "; R; " DATASETS DISPLAYED AND FILE CLOSED."
600  END
```

*Adapted from Data File Programming in BASIC by LeRoy Finkel and Jerald R. Brown, pages 258 and 262; copyright (c) 1981 by John Wiley and Sons, Inc., New York, New York. Reproduced with permission.

NAME _____
DATE _____
PERIOD _____

RANDOM ACCESS DATA FILES*

3. Write a program to display the contents of selected records from the random access disk file named PRODUCT that was created in problem #1 and used in problem #2.

Write a second program that will allow the user to select a record to be displayed from the sequential file, POINT, created in problem #2.

A flowchart and good documentation are required.

```
110 REM
120 REM  VARIABLES USED
130 REM    P$ = PROD. NO. (4)
140 REM    I$ = ITEM DESCRIPTION(20)
150 REM    S$ = SUPPLIER(20)
160 REM    L  = REORDER POINT
170 REM    Y  = REORDER QUANTITY
180 REM    Q  = QUANTITY
190 REM    C  = WHOLESALE COST
200 REM    U  = UNIT SELLING PRICE
210 REM    R$ = PRESS 'ENTER' TO CONTINUE VARIABLE
220 REM
230 REM  FILE USED
240 REM    PRODUCT = (RANDOM ACCESS FILE)
250 REM
```

Possible Solutions to Problem 3
Using Applesoft BASIC

```
100  REM  PROB 3 SOLUTION USING THE RANDOM ACCESS FILE, PRODUCT
110  REM
120  REM  VARIABLES USED
130  REM    P$ = PROD. NO. (4)
140  REM    I$ = ITEM DESCRIPTION (20)
150  REM    S$ = NAME OF SUPPLIER (20)
160  REM    L  = REORDER POINT (4)
170  REM    Y  = REORDER QUANTITY (4)
180  REM    Q  = QUANTITY IN STOCK (4)
190  REM    C  = WHOLESALE COST (4)
200  REM    U  = UNIT SELLING PRICE (4)
210         R  = SELECTED RECORD
220  REM
230  REM  FILE USED
240  REM    PRODUCT - (RANDOM ACCESS FILE)
250  REM
260  REM    INITIALIZE ROUTINE
265  CLEAR
270  D$ = CHR$(4)
275  ONERR GOTO 400
280  PRINT "ENTER A NEGATIVE NUMBER"
285  PRINT "TO STOP THE PROGRAM."
290  PRINT D$;"OPEN PRODUCT,L64"
300  INPUT "WHICH RECORD?";R
310  IF R<=0 GOTO 440
315  REM
320  REM    READ AND DISPLAY THE SELECTED DATASET FROM PRODUCT
330  PRINT D$;"READ PRODUCT,R";R
340  INPUT P$ : INPUT I$
350  INPUT S$ : INPUT L
360  INPUT Y  : INPUT Q
370  INPUT C  : INPUT U
380  PRINT D$ : PRINT
390  PRINT P$;I$;S$;L;Y;Q;C;U
395  GOTO 300
400  PRINT "THE FILE IS NOT THAT LONG."
410  GOTO 290
420  REM
430  REM CLOSE FILE ROUTINE
440  PRINT : PRINT "ALL REQUESTED DATASETS FROM PRODUCT DISPLAYED
450  PRINT D$;"CLOSE PRODUCT"
460  PRINT "FILE CLOSED."
470  END
```

```
100  REM  PROB 3 SOLUTION USING THE SEQUENTIAL FILE, POINT
110  REM
120  REM  VARIABLES USED
130  REM    P$ = PROD. NO.
140  REM    Q  = QUANTITY IN STOCK
150  REM    R  = RECORD NUMBER
160  REM    N  = SELECTED RECORD
170  REM    I  = INDEX FOR LOOP
180  REM    R$ = USER RESPONSE TO CONTINUE
190  REM  FILE USED
200  REM    POINT - SEQUENTIAL FILE
210  REM
220  REM INITIALIZE ROUTINE
230  CLEAR
240  D$ = CHR$(4)
250  ONERR GOTO 340
260  PRINT D$;"OPEN POINT"
270  PRINT "SELECT A RECORD NUMBER."; N
280  FOR I = 1 TO N
290  PRINT D$;"READ POINT"
300  INPUT P$ : INPUT Q : INPUT R
310  IF I = N THEN PRINT P$, Q, R : GOTO 340
320  NEXT I
325  REM
330  REM CLOSE FILE ROUTING
340  PRINT D$;"CLOSE POINT"
350  INPUT "DO YOU WISH TO CONTINUE? (YES/NO)"; R$
360  IF LEFT$(R$,1) = "Y" THEN HOME : GOTO 270
370  END
```

*Adapted from Data File Programming in BASIC by LeRoy Finkel and Jerald R. Brown, pages 289 and 294; copyright (c) 1981 by John Wiley and Sons, Inc., New York, New York. Reproduced with permission.

NAME_____
DATE_____
PERIOD_____

CREATING AN INDEXED-SEQUENTIAL FILE

Sometimes a file (random or sequential) becomes too large to sort in the computer's limited main memory. You, therefore, need to create a smaller file, called an INDEX, which contains only the essential elements that you desire sorted, such as name or ID numbers, and the KEY or location of that record in the MASTER file. Once this indexed file is sorted it can, for example, be read sequentially to print the MASTER file in alphabetic order. Simply retrieve the KEY portion of the indexed record, and you have the location of the master record in MASTER file.

In order to create this indexed file, you must first place the essential element(s) that is/are to be sorted into an array. Why an array? Because you can use array expressions to do operations (such as sorts) on the entire group of numeric or character array data-items instead of on each data-item individually (scalar operations).

When an array is sorted, character arrays are sorted or indexed according to the collating sequence in effect on your computer, and numeric arrays are indexed or sorted numerically.

You may use any type of sort to alphabetically arrange the array elements. The sample solution for this problem contains a TRSDOS Mod III system sort command (CMD "O").

Once the array is sorted, each array element can be stored in a sequential file which can be used to retrieve and update records from the MASTER file.

*NOTE: The solution uses a variable length record for the random file, MASTER. Please be sure to enter the number of variable length files (2V) you are using when BASIC prompts you for "How Many Files?"

STUDENT WORKSHEET
Creating and Indexed-Sequential File

A. STATEMENT OF THE PROBLEM

Create a structured program that will print out an alphabetic listing of the random file, MASTER, through the use of an indexed-sequential file called INDEX.

The MASTER record looks like this:

	S$	N$	C$	D$
	SEX	STUDENT'S NAME	COURSE NO	OTHER DATA
1	M	MONIZ	0006	XXXXXXXXXXXXXXXXXXXXXXXXX
2	F	YUEN	0001	XXXXXXXXXXXXXXXXXXXXXXXXX
3	F	SMITH	0009	XXXXXXXXXXXXXXXXXXXXXXXXX
4	M	ABE	0006	XXXXXXXXXXXXXXXXXXXXXXXXX
5	F	KAMAKA	0002	XXXXXXXXXXXXXXXXXXXXXXXXX

FIELD	FIELD LENGTH
SEX	1 character
STUDENT'S NAME (LAST)	25 characters
COURSE NO.	4 characters
OTHER DATA	50 characters.

The output should look like this:

ALPHABETIC MASTER FILE LISTING

	STUDENT NAME	SEX	COURSE NO	OTHER DATA
1	ABE	M	0006	XXX
2	KAMAKA	F	0002	XXX
3	MONIZ	M	0006	XXX
4	SMITH	F	0009	XXX
5	YUEN	F	0001	XXX

The Indexed-Sequential file should look like this:

NAM$	RECLO
STUDENT NAME	RECORD LOCATION
ABE	4
KAMAKA	5
MONIZ	1
SMITH	3
YUEN	2

1. List the main objective(s) of this program.

2. What data are contained in each MASTER file record? What variable names will you assign to these elements? What is the length of each field?

3. What field(s) must be included in the array that will create the Indexed-Sequential file? Which is the primary field that must be first in the array element?

4. How will you create the elements of the above array?

5. What other item(s), if any, must be included in each array element?

6. Will the array be an alphabetic or numeric array? How large will you dimension this array?

7. What name will you assign to the Indexed-Sequential file?

A Possible Solution to the Problem of
Creating an Indexed-Sequential File
for an Alphabetic Master Student List

```
005 REM WRITTEN FOR TRS-80 MODEL 3 AND 4 COMPUTERS
010 REM VARIABLE NAMES USED
020 REM    MASTER = MASTER FILE
030 REM    INDEX  = INDEXED-SEQUENTIAL FILE
040 REM    N$     = NAME OF STUDENT (25)
050 REM    S$     = SEX (1)
060 REM    C$     = COURSE (4)
070 REM    D$     = OTHER DATA (50)
080 REM    N%     = NUMBER OF RECORD BEING READ (INTEGER)
090 REM    NAM$   = NAME FIELD INDEXED-SEQUENTIAL FILE
100 REM    RECLO  = RECORD LOCATION ON MASTER FILE
110 REM    RL     = LENGTH OF MASTER FILE RECORD - 80 CHARACTERS
120    CLEAR 30000
130    DIM A$(100)
135    RL = 80
137    N% = 0
140    OPEN "R",1,"MASTER",RL
150    OPEN "O",2,"INDEX"
160    FIELD 1, 1 AS S$, 25 AS N$, 4 AS C$, 50 AS D$
180    IF EOF(1) THEN N% = N% - 1
             ELSE GOSUB 2000:
                  GOTO 180
200 REM * * * * * * * * * * * * * * * * * * * * * * * * * * * *
210 REM              SORTING THE ARRAY A$(N%)
220 REM * * * * * * * * * * * * * * * * * * * * * * * * * * * *
230    CMD"O", N%, A$(1)
300 REM * * * * * * * * * * * * * * * * * * * * * * * * * * * *
310 REM     CREATING INDEX FROM SORTED ARRAY A$(N%)
320 REM * * * * * * * * * * * * * * * * * * * * * * * * * * * *
340    FOR I = 1 TO N%
350          PRINT #2, A$(I)
360          PRINT A$(I)
370    NEXT I
380    CLOSE 2
400 REM * * * * * * * * * * * * * * * * * * * * * * * * * * * *
410 REM     PRINTING THE ALPHABETICAL LISTING OF MASTER FILE
420 REM * * * * * * * * * * * * * * * * * * * * * * * * * * * *
421    LPRINT CHR$(12): LPRINT TAB(40) "ALPHABETIC MASTER FILE LISTING"
423    LPRINT: LPRINT "  NAME"; TAB(30)"SEX"; TAB(47)"COURSE", "OTHER DATA"
425    LPRINT:LPRINT
430    OPEN "I",2,"INDEX"
440    FOR I = 1 TO N%
450          INPUT #2,NAM$,RECLO
460          GET 1, RECLO
470               LPRINT N$,S$,C$,D$
480    NEXT I
```

```
0500 REM * * * * * * * * * * * * * * *
0510 REM     END OF JOB ROUTINE
0520 REM * * * * * * * * * * * * * * *
0540    CLOSE
0550 END
2000 REM * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
2010 REM     CREATING ARRAY A$(N%) FROM MASTER FILE RECORDS
2020 REM * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
2040    N% = N% + 1
2050    GET 1, N%
2060            A$(N%) = N$ + "," + STR$(N%)
2070            PRINT A$(N%)
2080    RETURN
```

NAME _____
DATE _____
PERIOD _____

TERM PROJECT

The following is a list of the items you will need for your term project to receive credit. The project is due on _____. Extra points will be added for projects handed in early and points will be subtracted for late projects.

1. A cover of some sort with your name and project title on it.
2. A detailed written description of the problem of your program will "solve" and the procedure that you will use to solve it. Include any equations that must be used.
3. Operating instructions for running your program.
4. List of variables used in your program and their purpose. This list may be in alphabetical order or the order in which they are used in your program.
5. A flowchart or algorithm.
6. A listing of the program and a sample run, if possible.
7. The grading sheet.

Other items that can be included in your project for additional credit are:

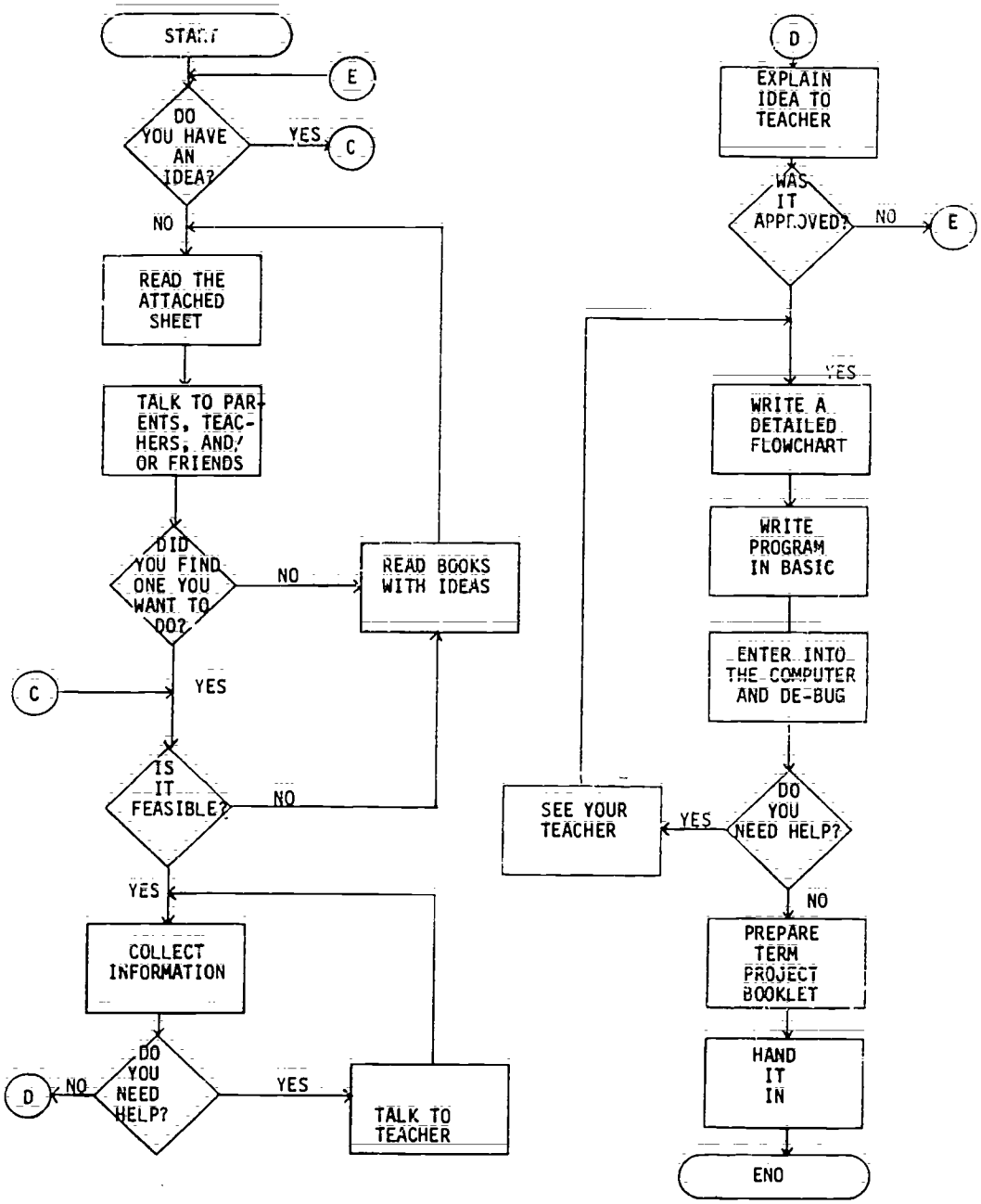
1. A hierarchy chart;
2. A logo for the program, using graphics;
3. A "menu" to make the program more user friendly.

TERM PROJECT
IDEAS

CONVERTING FROM DECIMAL TO BASES OTHER THAN BINARY
SOLVING SYSTEMS OF EQUATIONS
MULTIPLE CHOICE TEST IN ANY SUBJECT
SOLVING TRIANGLES USING TRIGONOMETRY
GRAPHING LINEAR, QUADRATIC, POLAR EQUATIONS
PASCAL'S TRIANGLE
MAGIC SQUARES
PRIME NUMBERS
MUSIC
METRIC CONVERSIONS
PALINDROMES
SORTING METHODS
GAMES, SUCH AS:
TIC TAC TOE, DICE GAMES, JAN KEN PO, REVERSE, ACEY DUECY, BINGO, BURIED TREASURE
ADDRESS BOOK
DRAWING PROGRAM (ETCH A SKETCH)
HAWAII STATE DRIVER'S TEST
ANAGRAMS
SIMULATIONS (SLOT MACHINE)
CONIC SECTIONS
CHECKBOOK PROGRAM
GIVEN A YEAR, PRINT THE CALENDAR FOR THAT YEAR
TUTORIAL PROGRAM (TEACHING RULES OF MULTIPLYING WITH SIGNED NUMBERS)
MAD LIBS
PROGRAMS FOR TEACHERS, DEPARTMENTS OR CLUBS, SUCH AS:
TEST BANKS, SURVEY RESULTS, TEXT/EQUIPMENT INVENTORIES, SCHOOL PAPER, YEARBOOK

OR ANY IDEAS OF YOUR OWN (SUBJECT TO TEACHER APPROVAL!) USE OF TEXT FILES IS ENCOURAGED.

TERM PROJECT FLOWCHART



NAME _____
DATE _____
PERIOD _____

TERM PROJECT
GRADING SHEET

Project Title _____

	Possible	Points
Neatness, appearance, organization	40	_____
Description of problem and solution	30	_____
Topic difficulty	40	_____
Topic suitability, usefulness, and creativity	30	_____
Flowchart or algorithm	30	_____
List of variables	30	_____
Program Does it work? Does it check all possibilities of data? Is it user friendly with a nice screen display?	100	_____
TOTAL	300	_____

Date turned in _____

Early/Late points _____

FINAL TOTAL _____

1 point for every school day it is early (Maximum of ___ points)
5 points for every school day it is late

SAMPLE ACTIVITY CLUSTER #6
Data Structures

Topics:

1. Stacks
2. Queues and Linked Lists
3. Trees

Classroom Management:

The class as a group will be involved with most of these activities. Some assignments may require pairs of students.

Materials:

Textbook, Data Structures for Personal Computers by Y. Langsam, M. J. Augenstein and A. M. Tenenbaum;
Videotape player and videotape, "Database Processing Systems"
Overhead projector and prepared transparencies;
Handouts.

Time for Activities:

Approximately one and a half to three weeks may be needed, depending on the level of skills of the students. Time for the term project should also be provided.

Teacher Preparation and Procedures:

Prepare needed transparencies and handouts. Preview videotape, if possible. All sample materials for these activities are located in the Sample Assignments and Materials for Activity Cluster #6.

1. Using the overhead projector, introduce the major concepts of data structures. Distribute Handout #1 - Outline Notes. Start with the easiest--the stack--an ordered collection of items into which new items may be inserted and from which items may be deleted at one end. Provide illustrations.

2. Assign some reading and/or exercises related to stacks for pairs of students to do. Handout #2 - Exercises for the Stack Data Structure could be used. When students have finished, discuss and compare results.

3. Continue the lecture on data structures by discussing queues and linked lists. Distinguish the difference between stacks and queues. (Both are lists of things, but queues have one end for adding items and another end for removing items.) Linked lists have pointers indicating the beginning, the end, and other areas in the list. Provide illustrations. Refer to Handout #1.

4. Assign some reading and/or exercises related to queues and linked lists for pairs of students to do. Handout #3 - Queue Problems and Handout #4 - Exercises for the Linked List Data Structure could be used. Have students discuss and compare results when finished.

5. Finalize the lecture on data structures by presenting tree structures, especially binary trees. Emphasize the difference in searching techniques between trees and lists. (Linked lists require a linear search, whereas binary trees allow binary search techniques.) Provide illustrations. Refer to Handout #1.

6. Assign some reading and/or exercises related to binary trees for pairs of students to do. Handout #5 - Exercises for the Binary Tree Data Structure can be used. Again, have students discuss and compare results when finished.

7. Show the videotape, "Database Processing Systems," which contrasts file processing with database processing. Tree structures play a major role in databases and are briefly discussed. Provide Handout #6 - Questions for the Videotape, "Database Processing Systems" for student discussion following the presentation.

8. Use discretion as to the appropriateness in assigning problems that incorporate data structures in BASIC programming. Time and student level of comprehension may limit such activities. The textbook suggested on the previous page will be most helpful here.

Sample Assignments and Materials for Activity Cluster #6:

The following pages in this activity cluster provide handouts that could be used in this course.

NAME _____
DATE _____
PERIOD _____

OUTLINE NOTES

Data Structures

I. Stack - an ordered collection of items into which new items may be inserted and from which items may be deleted at one end--the "top."

A. Examples -

1. A stack of dishes in a cafeteria that supplies the top dish to the next customer; only the top dish can be removed.
2. A stack of magazines.
3. A stack of business records in a file.

B. LIFO Structures

1. Last In, First Out; last item put on stack, first to be removed.
2. Illustratration of four different views of the same stack. (See Illustration 1 which follows the outline.)

C. Primitive Operations--given a stack 's' and an item 'i'

1. Push - to place something on the top of the stack: push(s,i).
2. Pop - to remove something from a stack: pop(s).
3. Empty - to determine whether or not a stack is empty: empty(s).
4. Stacktop - to determine the top item on a stack without removing it: stacktop(s); this cannot be applied to an empty stack, as it is equivalent to pop(s) and push(s,i).

D. Problem: check all parentheses and brackets for matches to see if this string is valid - (x + (y-[a + b]) * c - [(d + e)])/(h - (j-(k-[l - n]))).

1. Algorithm -

```
Set validity flag to true.
Set stack(s) to empty.
Do until flag is false or entire string has been read

    Read next symbol(i) in string
    If symbol is '(', '[', '  ' ("openers") then push(s,i)
    Else
        If symbol is ')', ']', '  ' ("closers") then
           If empty(s) then set flag to false
           Else
```

```
                pop(s,i)
                If i is not the matching "opener" for symbol
                   set flag to false
                Else
                   continue
            Else
               continue.
        If not empty(s) then set flag to false.
        If flag is valid then write "The string is valid."
        Else
            write "The string is invalid.".
```

2. Refer to Illustration 2.

3. Practice Problems; refer to Handout #2.

II. Queues and Linked Lists

A. Queue - an ordered collection of items from which items may be deleted at one end (the front) and into which items may be inserted at the other end (the rear).

1. Example of queues of people standing in line waiting for something; consider "sociology" of these kinds of queues.

2. Example of records in Illustration 3.

3. FIFO structures - first in, first out - as opposed to stacks.

4. Primitive Operations - given a queue 'q' and an item 'x':

   a. Insert - to add something at the rear of the queue: insert (q,x);
   b. Remove - to delete something from the front of the queue this cannot be applied to an empty queue: x=remove(q);
   c. Empty - to determine whether or not the queue is empty: empty(q);
   d. Refer to Illustration 3 again.

5. Underflow - result of an illegal attempt to remove an item from an empty queue.

6. Overflow - result of having more items or elements in an array, used as a queue, than were allocated for the array.

7. Specifications -

   a. Consider an array that holds a queue as a circle, not a straight line;

b. The value of the variable for the front element (FRONT) should be the index of the array element immediately preceding the first element of the queue;
c. The value of the variable for the last element (REAR) should be the index of the last element of the queue;
d. A queue can grow only as large as one less than the size of the array.

8. Illustrations of the specifications -

a. If an array of 5 positions is considered a queue, the queue may have up to 4 members or elements.
b. Examples of a full queue:

FRONT = 1 and REAR = 5 or
FRONT = 2 and REAR = 1

c. Refer to Illustrations 4 and 5.

9. Case Study Problem; refer to Handout #3.

B. Linked List - an ordered list consisting of elements called nodes, each having a data portion and a pointer portion that points to or gives the address of the next node. A "head" pointer points to the first element of the list, whereas a "nil" is the last pointer in the list. Refer to Illustration 6.

1. A dynamic data structure, not having a fixed amount of storage, as allocated to stacks or queues.

2. Can have an element easily inserted in or deleted from the middle whereas arrays in stacks or queues require vast movement of elements to accommodate the insertion or deletion. Refer to Illustration 7.

3. Operations - give 'p' as a pointer, 'x' as an item and node (p) as the node pointed to by p:

a. Insafter(p,x) - to insert an item into a list after a node pointed to by p;
b. Delafter(p,x) - to delete an item from a list following node(p);
c. Push and pop are still valid operations.

4. Practice Problems; refer to Handout #4.

III. Trees - a type of multiply-linked list, the nodes of which have more than one pointer field; a Binary Tree has no more than two branches (pointers) from each node.

A. Root - topmost element or node of the tree; organization is from top down.

B. Branch - a pointer field of a node in a tree.

C. Leaf - any node with both pointer fields equal to Nil.

D. Parent nodes point to children nodes below; every node is really the root of its own subtree. Refer to Illustration 8.

E. Terms or descriptions to define and identify on Illustration 8:

1. Strictly binary tree - every nonleaf node in a binary tree has non-empty right and left subtrees;

2. Complete binary tree of level 'n' - each node at level n is a leaf and each node at lesser levels has nonempty right and left subtrees;

3. Almost complete binary tree - leaves are found only in the last two levels of a binary tree;

4. Level - the root of a tree is at level 0, and the level of any other node is 1 more than the level of its parent;

5. Depth - the maximum level of any leaf in the tree.

F. Operations on Binary Trees - given 'p' as a pointer to a node 'nd':

1. Info(p) - to return contents of nd;

2. Left(p) - to return pointer to left child of nd;

3. Right(p) - to return pointer to right child of nd;

4. Parent(p) - to return pointer to parent of nd;

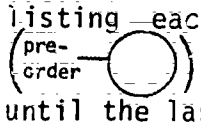
5. Sibling(p) - to return pointer to sibling of nd;

6. Maketree(x) - to create a new binary tree with one node having field x;

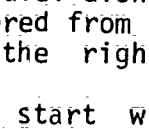
7. Setleft(p,x) - to accept pointer p to a binary tree node nd with no left child and create a left child of nd with data field x;

8. Setright(p,x) - to do the same thing as above but with a right child.

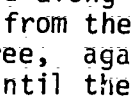
G. Traverse a binary tree - to pass through the tree, enumerating each of its nodes once:

1. Preorder - start at the root, travel along the left subtree, listing each node as it is encountered from the left side, 

, continue on to the right subtree, again listing each node as it is encountered from the left side until the last node is reached;

2. Inorder - start with the last left descendent on the left subtree, travel along the left subtree, listing each node as it is encountered from the bottom, 

, continue to the root and on to the right subtree in a similar manner;

3. Postorder - start with the bottom node on the left subtree, travel upward along the right side, listing each node as it is encountered from the right side, 

, continue on to the right subtree, again listing each node from the right side until the last node is reached at the root;

4. See Illustration 9.

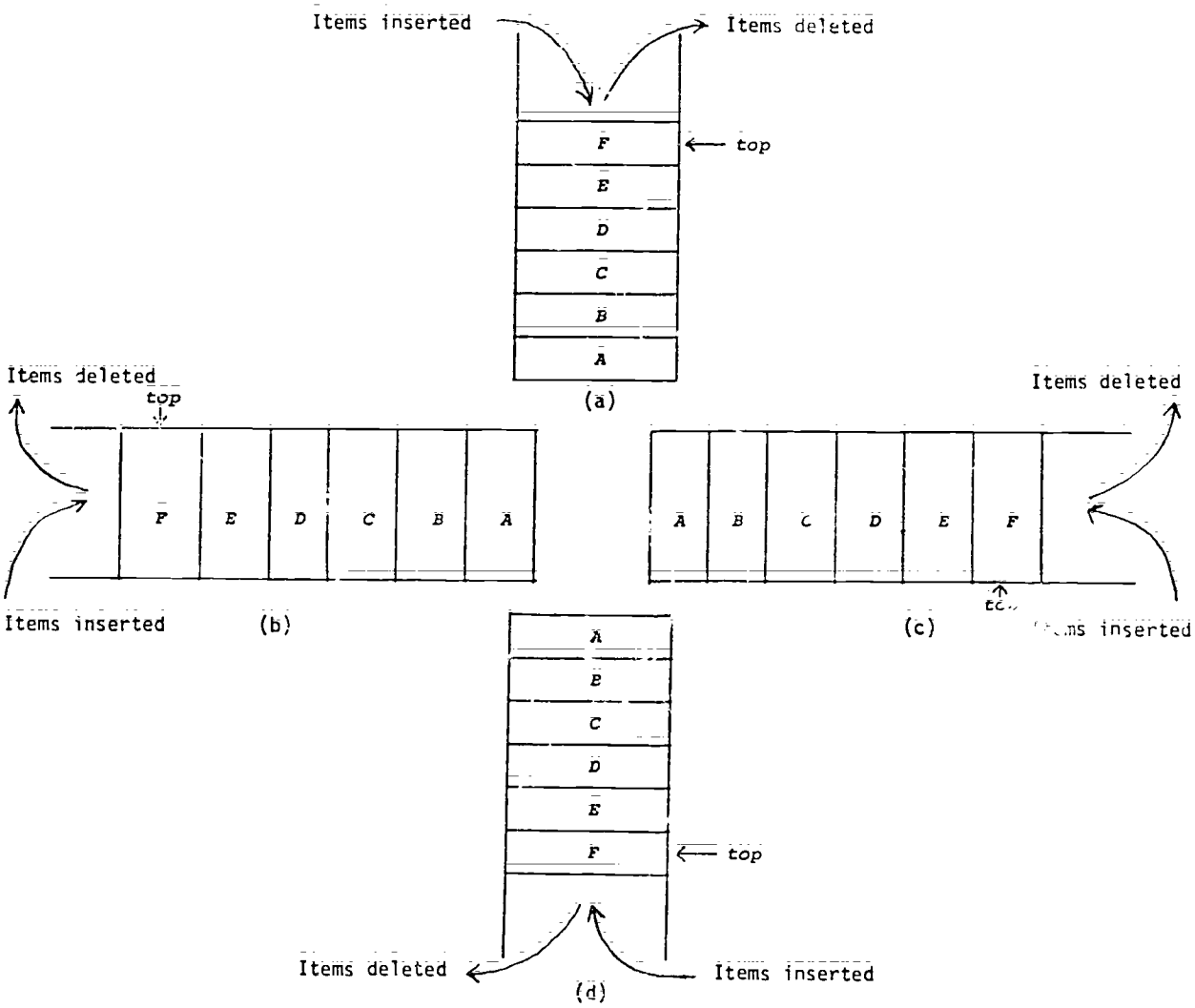
H. Example of an algorithm using the operation Intra(tree) to traverse a binary tree in inorder procedure and to print the contents of each of its nodes.*

```
Read number
tree = maketree(number)
while there are numbers left in the input do
    read number
    q = tree
    while q <> null do
        p = q
        if number < info(p)
            then q = left(p)
            else q = right(p)
        endif
    endwhile
    if number < info(p)
        then setleft(p, number)
        else setright(p,number)
    endif
endwhile
'traverse the tree
intrav(tree).
```

I. Practice Problems, refer to Handout #5.

*From Data Structures for Personal Computers by Langsam, Augenstein and Tenenbaum, page 294; copyright (c) 1985 by Prentice-Hall, Inc., Englewood Cliffs, New Jersey. Reproduced with permission.

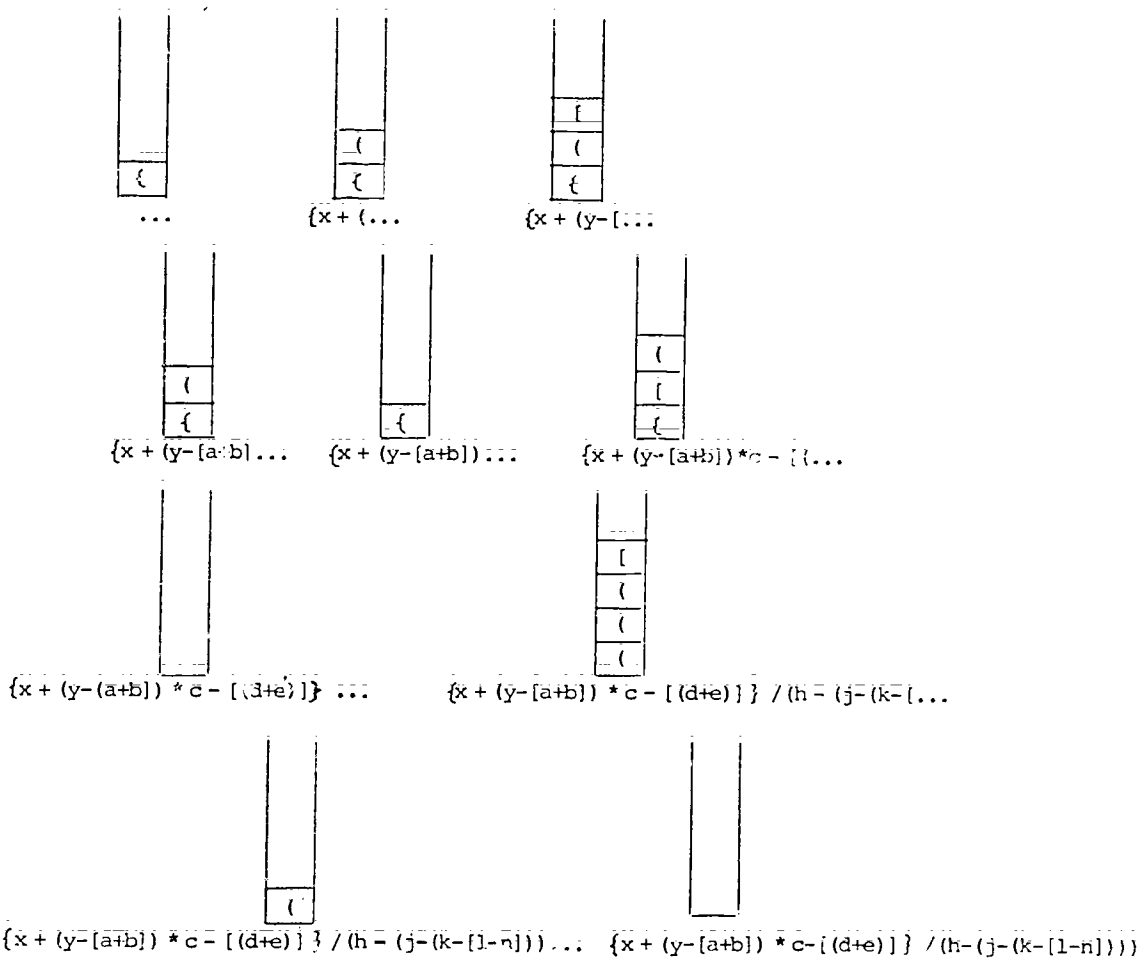
ILLUSTRATION 1
THE STACK*



Four different views of the same stack.

*From Data Structures for Personal Computers by Langsam, Augenstein and Tenenbaum, page 110; copyright (c) 1985 by Prentice-Hall, Inc., Englewood Cliffs, New Jersey. Reproduced with permission.

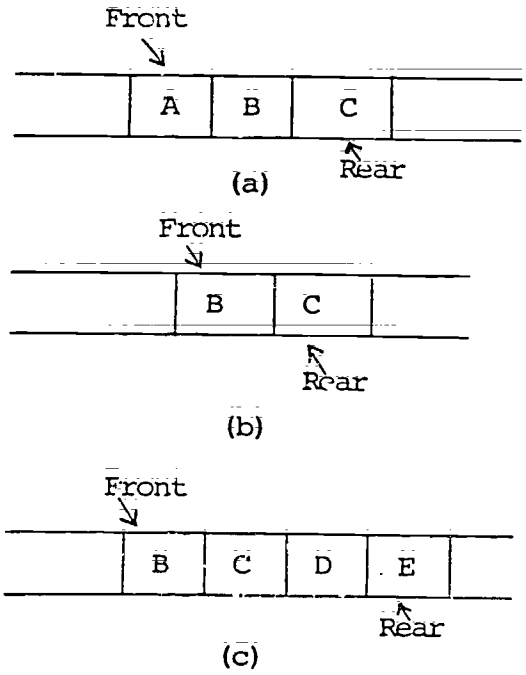
ILLUSTRATION 2
THE STACK*



The parenthesis stack at various stages of processing.

*From Data Structures for Personal Computers by Langsam, Augenstein and Tenenbaum; copyright (c) 1985 by Prentice-Hall, Inc., Englewood Cliffs. Reproduced with permission.

ILLUSTRATION 3
THE QUEUE*

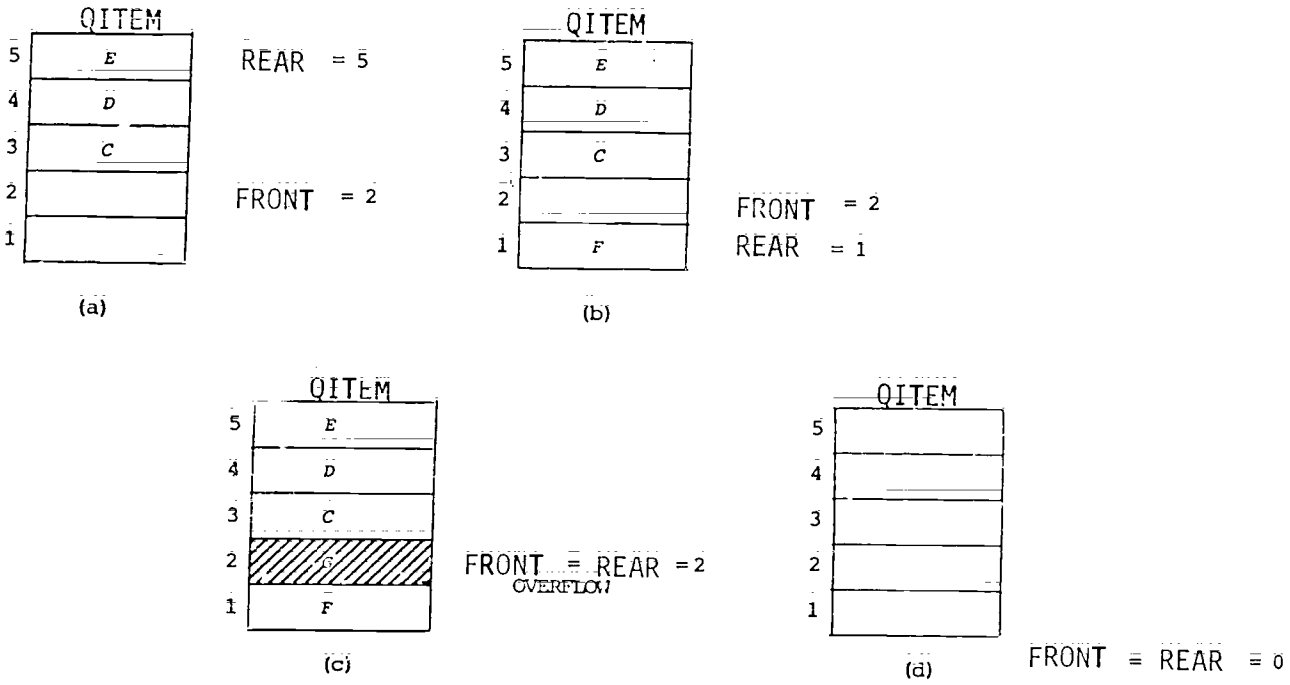



```
insert(q,A);
insert(q,B);
insert(q,C);          Figure (a)
x = remove(q);        Figure (b)
insert(q,D);
insert(q,E)           Figure (c)
```

Queues are essential to time-shared computer systems. They are used to keep track of each user's input. The computer executes commands at one end of the queue while adding new commands to the other end as they come in.

*From Data Structures for Personal Computers by Langsam, Augenstein and Tenenbaum, page 155; copyright (c) 1985 by Prentice-Hall, Inc. Englewood Cliffs, New Jersey. Reproduced with permission.

ILLUSTRATION 4
THE QUEUE*



Figure(a): there are three elements in the queue - C, D and E in QITEM(3), QITEM(4) and QITEM(5), respectively. Here REAR = 5 and FRONT = 2.

Figure(b): element F is inserted at the rear, thus moving REAR to position 1; according to given specifications of a queue, this one is full.

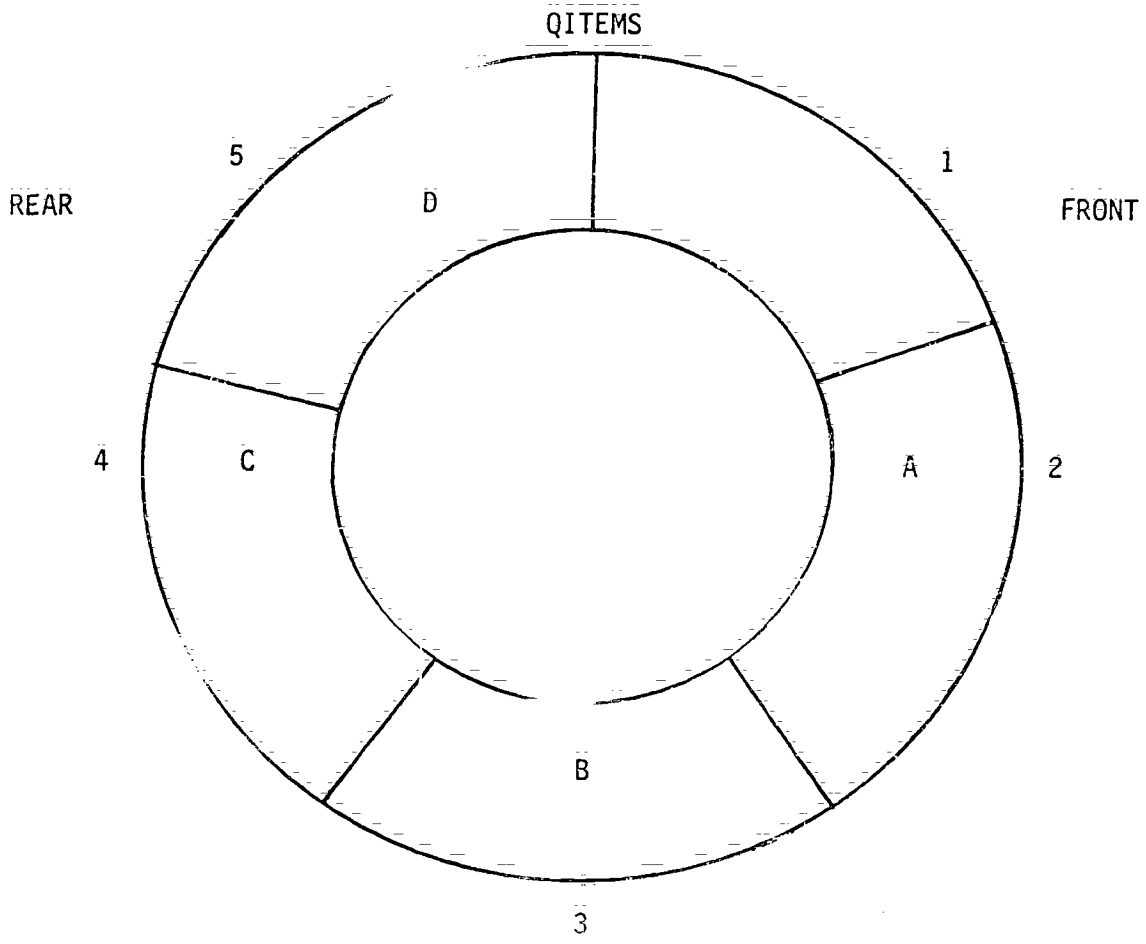
Figure(c): element G is inserted but cannot be accepted, since an overflow results when FRONT = REAR.

Figure(d): no elements mean an empty queue and FRONT = REAR = 0

*From Data Structures for Personal Computers by Langsam, Augenstein and Tenenbaum, page 161; copyright (c) 1985 by Prentice-Hall, Inc., Englewood Cliffs, New Jersey. Reproduced with permission.

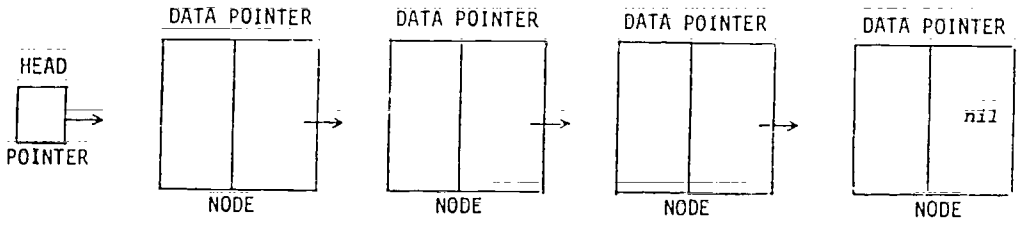
ILLUSTRATION 5
THE QUEUE

A queue simulated by a circular array



According to given specifications of a queue, this one is full.

ILLUSTRATION 6
THE LINKED LIST*



A linear linked list.

Each element in a linked list is known as a NODE and contains the DATA field with the information to be stored and the POINTER field with the address that points to the next element in the list.

*From Data Structures for Personal Computers by Langsam, Augenstein and Tenenbaum, page 165; copyright (c) 1985 by Prentice-Hall, Inc., Englewood Cliffs, New Jersey. Reproduced with permission.

ILLUSTRATION 7
THE LINKED LIST*

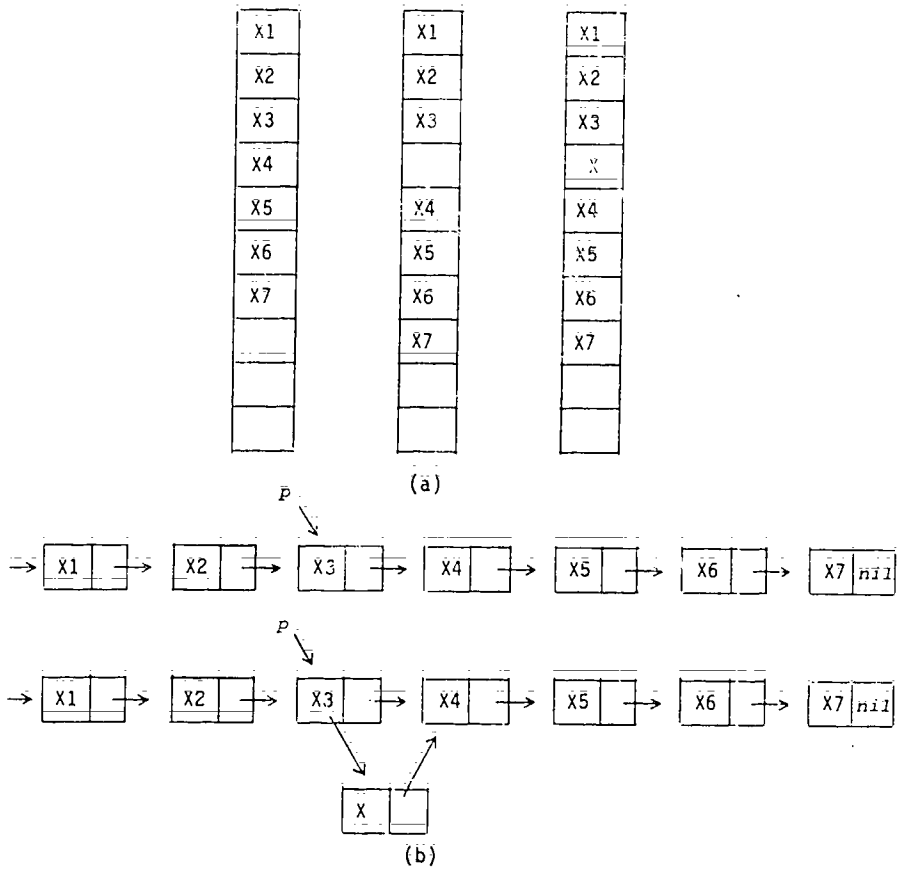
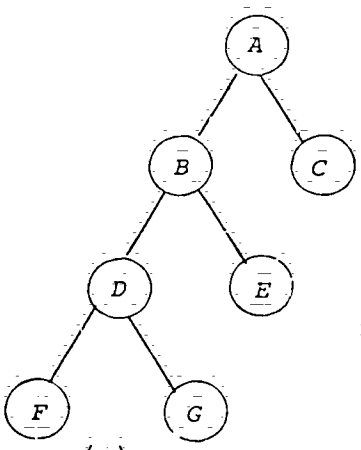


Figure (a): Inserting an element x between the third and fourth elements in an array or queue means moving items 4 through 7 each one slot first before the insertion. If the array had 1000 elements, a vast amount of changes would have to be made.

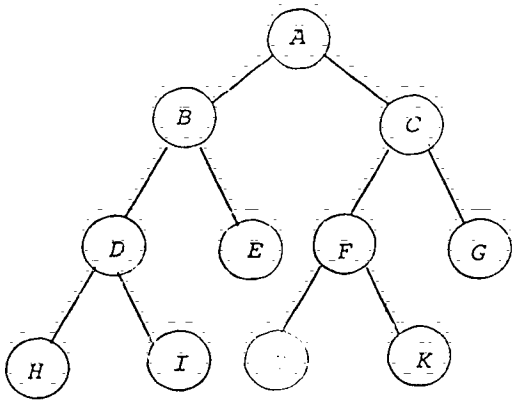
Figure (b): Inserting an element x between the third and fourth elements in a linked list means inserting the element and adjusting only two pointers. The amount of work required is independent of the size of the list.

*From Data Structures for Personal Computers by Langsam, Augenstein and Tenenbaum, pages 175-176; copyright (c) 1985 by Prentice-Hall, Inc., Englewood Cliffs, New Jersey. Reproduced with permission.

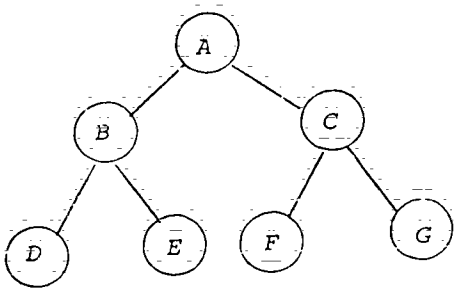
ILLUSTRATION 8
THE BINARY TREE*



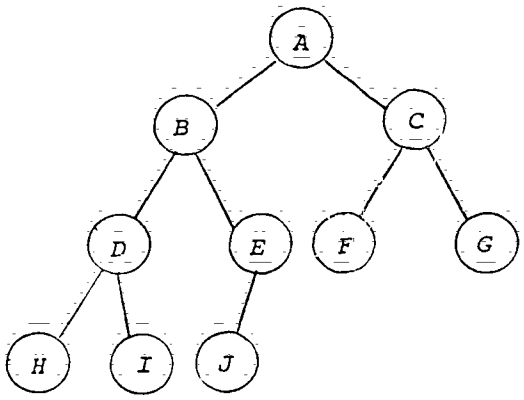
(a)
A Strictly Binary Tree



(b)
A Strictly Binary Tree

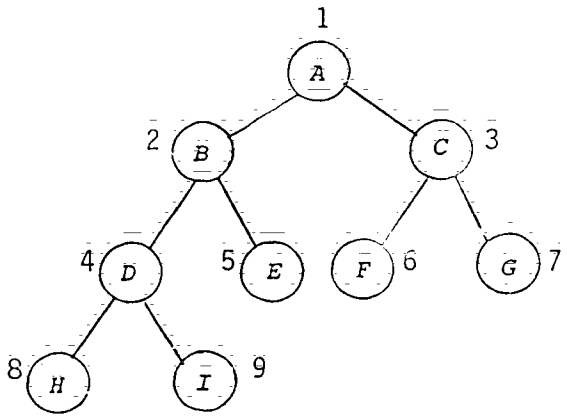


(c)
A Strictly Binary Tree and
A Complete Binary Tree of level 2



(d)

An Almost Complete Binary Tree



(e)

A Strictly Binary Tree (with proper numbering of nodes)

*From Data Structures for Personal Computers by Langsam, Augenstein and Tenenbaum, page 288; copyright (c) 1985 by Prentice-Hall, Inc., Englewood Cliffs, New Jersey. Reproduced with permission.

ILLUSTRATION 9
THE BINARY TREE*

Binary Trees and Their Traversals:

To traverse a nonempty binary tree in preorder:

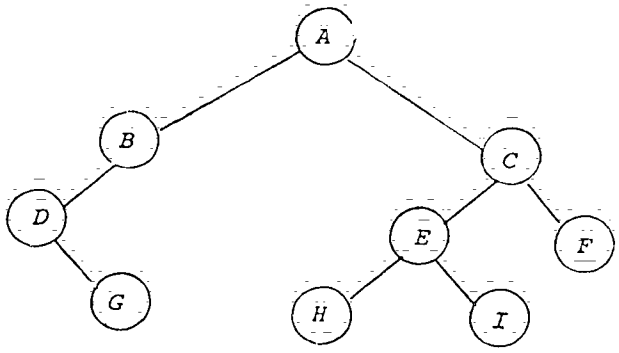
1. Visit the root.
2. Traverse the left subtree in preorder.
3. Traverse the right subtree in preorder.

To traverse a nonempty binary tree in inorder or symmetric order:

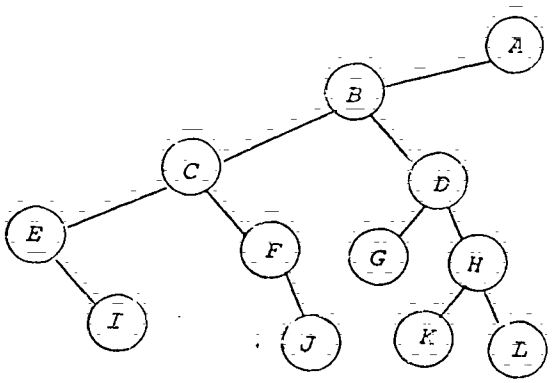
1. Traverse the left subtree in inorder.
2. Visit the root.
3. Traverse the right subtree in inorder.

To traverse a nonempty binary tree in postorder:

1. Traverse the left subtree in postorder.
2. Traverse the right subtree in postorder.
3. Visit the root.



Preorder: ABDGCEHIF
Inorder: DGBAHEICF
Postorder: GDBHIEFCA



Preorder: ABCEIFJDGHKL
Inorder: EICFJBGDKHLA
Postorder: IEJFCGKLHDBA

*From Data Structures for Personal Computers by Langsam, Augenstein and Tenenbaum, page 2; copyright (c) 1985 by Prentice-Hall, Inc., Englewood Cliffs, New Jersey. Reproduced with permission.

Handout #2
Data Structures

NAME_____
DATE_____
PERIOD_____

EXERCISES FOR THE STACK DATA STRUCTURE*

1. Use the operations push, pop, stacktop, and empty to construct operations which do each of the following:

   (a) Set i to the second element from the top of the stack, leaving the stack without its top two elements.
   (b) Set i to the second element from the top of the stack, leaving the stack unchanged.
   (c) Given an integer n, set i to the nth element from the top of the stack, leaving the stack without its top n elements.
   (d) Given an integer n, set i to the nth element from the top of the stack, leaving the stack unchanged.
   (e) Set i to the bottom element of the stack, leaving the stack empty.
   (f) Set i to the bottom element of the stack, leaving the stack unchanged. (Use another, auxiliary stack.)
   (g) Set i to the middle element from the bottom of the stack.

2. Simulate the operation of the algorithm in this section for each of the following strings by showing the contents of the stack at each point.

   (a) (a+b)
   (b) {[a+b] - [(c-d)]
   (c) (a+b) - {c+d} - [f+g]
   (d) ((h) * {([j+k])})
   (e) ((((a))))

3. Like the operation Pop, Stacktop is not defined for an empty stack. The result of an illegal attempt to Pop or access an item from an empty stack is called Underflow. Underflow can be avoided by ensuring that Empty(s) is false before attempting the operation Pop(s) or Stacktop(s).

   What set of conditions are necessary and sufficient for a sequence of push and pop operations on a single stack (initially empty) to leave the stack empty and not to cause underflow? What set of conditions are necessary for such a sequence to leave a non-empty stack unchanged?

*From Data Structures for Personal Computers by Langsam, Augenstein and Tenenbaum, page 117-118; copyright (c) 1985 by Prentice-Hall, Inc., Englewood Cliffs, New Jersey. Reproduced with permission.

NAME _____
DATE _____
PERIOD _____

QUEUE PROBLEMS
CASE STUDY*

Case Study: Simulation of a queue

Let's now look at an example of using a queue as a circular array.

For this case study, the problem is to simulate a single service station queue. We want to construct an algorithm which can be used to study what happens in the simplified situation where we have one clerk serving people who line up as they arrive and wait for service. This is the case in a small shop with only one clerk, or in a small post office, etc. In such a problem we would want to be able to watch the queue grow and shrink; we would want to know how much of the time the clerk was idle, and we would want to have some idea of the average time each customer stayed in the store --time spent waiting in the queue plus time getting served. We will want to be able to vary the average time between arrivals and the average service time and watch the effect these parameters have on the queue.

This is a simplified version of what might be done in an advanced course in simulation, and there is a great deal more that could be done; since our real goal is to demonstrate the use of queues, we'll not go into the problem any further than this.

Problem Definition:

Write an algorithm which will simulate a queueing situation with one queue and one server. The average time between arrivals and the average service time should be determined by the user. The algorithm should show the user how the queue changes during a work period as well as provide some statistics on the total idle time of the server, the average queue length, the maximum queue length and the average time each customer is in the store.

Solution:

The solution to the problem lies in the use of a queue to hold the customers as they wait to be served. We will need to have a random number generator in order to simulate the arrival and service times.

A description of the solution:

Each time a customer arrives, place him or her on the queue. Whenever the clerk finishes serving a customer, perform the necessary calculations for that customer, and, if another customer is waiting in line, take that customer off the queue and serve him or her. If there are no customers in line, adjust the idle time of the clerk appropriately.

An algorithm:

An algorithm for the solution could be written in several ways. One way is to use a big loop which loops each time something happens, either a customer arrives or leaves. An alternative is a loop which loops once for each of some fixed time interval.

We'll use a driving loop which loops once each minute of the working day, followed by another loop which will take care of any customers left in the queue when the store closes. We assume that the number of minutes in the working day is known - a constant - and assume that the user will enter the average time between arrivals and the average service time.

Leaving aside the details of initializing the queue and obtaining the data from the user, after some work you might arrive at an upper level algorithm something like:

```
loop once for each minute of the day:
    if a customer arrives
        then put the customer on the queue
    if the service station is not empty
        then subtract 1 from the service time left
    if the service station is empty
        then
            if the queue is not empty
                then process start-service
            else add 1 to idle-time
loop while there are customers left in the queue:
    subtract 1 from the service time left
    if the service station is empty
        then process start-service
calculate and display the results
```

This is fairly clear except that just what gets done when a customer arrives or leaves is not spelled out. It is also not clear what to put on the queue when a customer arrives. After thinking about it, you would likely conclude that the only thing needing to be put on the queue is the time of arrival. And, the average time the customer is in the queue is essentially the same as the average time the customer is in the store except that the latter time should be longer by the average service time. In other words, whether we capture statistics on the average time in the queue or the average time in the store doesn't really matter.

A lower level algorithm - solving some of these details - might look like:

```
START

initialize
print "Enter the average time between arrivals - "
enter average-arrival-time
```

```
print "Enter the average service time - "
enter average-service-time
loop for time set to 0 thru number-of-minutes-in-work-day
    if arrival
        then
            insert(time)
            if queue-length > max-length
                then set max-length to queue-length
    if service-time-left > 0
        then subtract 1 from service-time-left
    if service-time-left = 0
        then
            if not emptyqueue
                then
                   set arrival-time to delete
                   set number-served to number-served + 1
                   set total-waiting-time to total-waiting-time
                                          + (time - arrival-time)
                   set service-time-left to service-time
                else add 1 to idle-time
    add queue-length to total-length
    if time mod 15 = 0 then display-queue
    ( end of for loop )
set time to number-of-minutes-in-work-day
loop while not emptyqueue
    add 1 to time
    subtract 1 from service-time-left
    if service-time-left = 0
        then
            set arrival-time to dequeue or delete
            set number-served to number-served + 1
            set total-waiting-time to total-waiting-time
                                   + (time - arrival-time)
            set service-time-left to service-time
    add queue-length to total-length
    if time mod 15 = 0 then display-queue
    ( end of while loop )
set average-waiting to total-waiting-time/number-served
set average-queue-length to total-length/number-of-minutes-
                                        in-day
print "The average time between arrivals was ", average-arrival-time
print "The average service time was ", average-service-time
print "The total number served was ", number-served
print "The average time spent waiting was ", average-waiting
print "The total idle time for the clerk was ", idle-time
print "The maximum queue length was ", max-length
print "The average queue length was ", average-length

END OF THE ALGORITHM
```

Illustration of Queue:

Draw an illustration of the queue in the above algorithm. Show an example at the beginning of the day, in the middle and at the end of the day. Using the recommended textbook, convert the above algorithm into BASIC code.

*Samuel E. Rhoads, Course Guide for Advanced Placement Computer Science, Department of Education, State of Hawaii, (1985), pages XIII-4 to XIII-7.

NAME_____
DATE_____
PERIOD_____

EXERCISES FOR THE QUEUE DATA STRUCTURE*

1. What set of conditions is necessary and sufficient for a sequence of insert and remove operations on a single empty queue to leave the queue empty without causing an underflow? What set of conditions is necessary and sufficient for such a sequence to leave a nonempty queue unchanged?

2. If an array is not considered circular, it is suggested that each remove operation must shift down every remaining element of a queue. An alternative method is to postpone shifting until REAR equals the last index of the array. When that situation occurs and an attempt is made to insert a element into the queue, the entire queue is shifted down so that the first element of the queue is in the first position of the array. What are the advantages of this method over performing a shift on each remove operation? What are the disadvantages? Rewrite the routines remove, insert and empty using this method.

3. Show how a sequence of insertions and removals from a queue represented by a linear array can cause an overflow to occur upon an attempt to insert an element into an empty queue.

Handout #4
Data Structures

NAME _____
DATE _____
PERIOD _____

EXERCISES FOR THE LINKED LIST DATA STRUCTURE*

1. Write an algorithm for inserting a new element to a linked list at the beginning of it. Illustrate this processing using blocks and arrows. Don't forget the Head Pointer; be sure to label all nodes.

2. Write an algorithm for inserting a new element at the end of a linked list. Illustrate this processing using a method similar to above.

3. Write an algorithm for the following:

   a. Concatenate two lists;
   b. Reverse a list, so that the last element becomes the first, and so on;
   c. Delete the last element from a list;
   d. Delete the nth element from a list;
   e. Insert an element after nth element of a list
   f. Delete every second element from a list.

4. Write a BASIC routine to perform one of the above algorithms.

*From Data Structures for Personal Computers by Langsam, Augenstein and Tenenbaum, pages 187-188 copyright (c) 1985 by Prentice-Hall, Inc., Englewood Cliffs, New Jersey. Reproduced with permission.

Handout #5
Data Structures

NAME_____
DATE_____
PERIOD_____

EXERCISES FOR THE BINARY TREE DATA STRUCTURE*

1. Prove that a node of binary tree has at most one parent.

2. What are the maximum number of nodes at level 'n' in a binary tree?

3. Write an algorithm to determine if a binary tree is:

   a) strictly binary;
   b) complete;
   c) almost complete.

4. Two binary trees are similar if they are both empty or both nonempty, their left subtrees are similar and then right subtrees are similar. Write an algorithm that determines if two binary trees are similar.

*From Data Structures for Personal Computers by Langsam, Augenstein and Tenenbaum, pages 296-297; copyright (c) 1985 by Prentice-Hall, Inc., Englewood Cliffs, New Jersey. Reproduced with permission.

Handout #6
Data Structures

NAME _____
DATE _____
PERIOD _____

QUESTIONS FOR THE VIDEOTAPE,
"Database Processing Systems"

1. What is meant by Integrated Data Systems?
2. What are three disadvantages of a file processing system?
3. What role does the DBMS (Database Management System) play?
4. Consider five advantages and disadvantages of database processing.
5. What data structures are utilized for database data?
6. How is a subschema different from a schema in viewing database data? Give an example of each.
7. What is meant by overhead data? How does overhead data affect the system hardware and software?
8. How is the integrity of the database maintained?
9. What are the responsibilities of the Database Administrator?
10. What qualifications should a Database Administrator have?

SAMPLE ACTIVITY CLUSTER #7
Computer Ethics and Impact on Society

Topics:

1. Computer Careers
2. Computer Applications in Various Fields
3. Computer Misuses and Crime
4. Unemployment and Depersonalization
5. Privacy

Classroom Management:

Most of the activities will involve the whole class. Reading assignments will be done by the individual students.

Materials:

Resources for reference or text, such as:
- An Introduction to Computers by Donald D. Spencer, Chapters 7 and 16;
- Computer Literacy--Problem-Solving with Computers by C.E. Horn and J. L. Poirot, pages 98-107;
- Computer Literacy - Programming, Problem Solving, Projects on the Apple by Warren and Bobbie Jones, Kevin Bowyer and Mel Ray, Chapter 9;
- Computers Today by Donald H. Sanders, Chapter 19;
- Scholastic Computing--An Introduction to Computers by Jack L. Roberts, Unit 5;
- Spotlight on Computer Literacy by Ellen Richman, Chapter 12;
- "To Copy or Not to Copy: A Moral Dilemma" by Tim Barry from InfoWord, (September 29, 1980), pages 5-6;
- "Washington Tackles the Software Problem" by Christopher Kern from BYTE, (May 1981), pages 128-138;
- "Copying Software--Crime in the Classroom?" by Lauren Letellier from Electronic Learning, (January/February 1982), pages 42-51;
- "Teaching Computer Ethics" by William J. Kreidler from Electronic Learning, (January 1984), pages 54-57;
- Current articles from newspapers or magazines related to computers in society;
- Handouts.

Time for Activities:

Approximately one to two weeks

Teacher Preparation and Procedures:

Select appropriate reading material, such as from the above resources. Prepare needed handouts.

1. Ask students to be "on the lookout" for newspaper articles relating to the impact of computers on society and jobs and bring these in for sharing and posting.

2. Invite as guest speakers two or three professionals in the field of computing (programmer, systems analyst, database administrator, etc.) to discuss job responsibilities, problems and rewards.

3. Have students contact employment agencies, colleges and universities for data on particular computer-related jobs. Ask them to investigate job descriptions, the average salary and the courses or degrees required. Suggest they make use of the Computer Careers Handbook, an ARCO Publication, by Connie Winkler, found in the vocational department in most large libraries. Also, the Readers' Guide to Periodical Literature and Career Kokua may prove to be helpful.

   Allow time for students to share their research findings orally and to assemble their data on a bulletin board, along with advertisements, articles, want ads or other related resources.

4. Computer ethics is a topic that can be incorporated at any time in the semester, not just toward the end of the course, as suggested here. An appropriate time would be when an issue arises in class over copying commercial software or when the tampering of a student's private disk has occurred. A class discussion of the specific issue or the general topic should be conducted. Student responsibility for upholding high ethical standards in and out of the classroom cannot be overemphasized. After reviewing in class issues of particular news articles related to computers in society, encourage students to discuss the articles with their parents. Following these family discussions, have students share their reactions to and opinions about the articles. Input from parents, teacher and class can provide a broader perspective on major issues.

5. The topic of computer ethics can be further expanded with an article or two on computer crime. Have students read copies of the article(s) and discuss the basic issues, such as:

   a. Reasons why it is difficult to prevent computer crime;
   b. Ways in which computer crime affects the economy;
   c. Possible solutions for or preventative measures against computer crime;
   d. Pros and cons of the current copyright laws on software;
   e. Suggestions for a fair and just resolution of the dilemma.

6. Assign some reading material from a text or reference with further discussion questions.

7. Invite a guest speaker from the data processing or computer department in a large firm to come and address the issues of data privacy and security and computerized databases.

8. As an alternative approach to covering the topics, have teams of students research one particular topic, such as computer fraud or computer victims. Have students use a word processor to type up their reports. Then allow time for class presentations and discussions of the topics. A list of interesting topics related to computers in our society is provided in Handout #1 - The Impact of Computers on Our Society. A format for organizing research information and personal reactions can be found in Handout #2 - Summary Format.

NAME_____
DATE_____
PERIOD_____

THE IMPACT OF COMPUTERS ON OUR SOCIETY

Look over the following topics and subtopics related to computers and choose ONE of the subtopics that your team will develop for presentation to the class.

I. ROBOTICS AND AUTOMATION
   A. Artificial Intelligence--what is it? Can it be created?
   B. Robotics in the work force--how have robots affected manufacturing? List problems and advantages to us.
   C. What kinds of jobs will computers replace? What are the problems and advantages to the people they replace?

II. DATA BANKS
   A. What are data banks? What are they used for? How are they created? How is the information in them retrieved?
   B. What are our rights to privacy? How can we protect ourselves from wrong information in our files? Who has access to information on us? Where are these data banks located?
   C. What are some of the abuses that have occurred with data banks? What are some of the advantages of data banks?

III. COMPUTERS IN FINANCIAL INSTITUTIONS
   A. How do banks use computers? Why do they need them?
   B. What is Electronic Funds Transfer and how does it work? at are the advantages of its use? Disadvantages?
   C. What are some of the dangers that occur with the use of computers in banks? How can these dangers be minimized?

IV. CRIME AND LAW ENFORCEMENT
   A. What is a hacker? Who are hackers? What are some of the things that hackers have done? How are they found?
   B. What kinds of crimes can be committed by computers? How can they be prevented? How can they be detected?
   C. What are some of the ways in which computers are used in la enforcement?

V. COMPUTERS IN ENTERTAINMENT
   A. Video games--do they have any useful purpose besides entertainment? Are they dangerous or harmful? What are some of the latest developments in this field?
   B. How do movies use computers to create special effects?

VI. COMPUTERS IN EDUCATION
- A. What is Computer Assisted Instruction (CAI)? How can the schools use CAI to improve learning and teaching?
- B. How will computers change the way education is taught? Will we need schools at all?

VII. COMPUTERS IN BUSINESS
- A. How are computers used in sales? What do point-of-sales terminals do in a store? How do they help management?
- B. What is word processing? How can it be used in today's office? What are the advantages? Disadvantages? Will it eliminate many clerical/secretarial jobs?
- C. How are computers used in accounting? What is accounting? How can computers help with budgeting?

VIII. COMPUTERS AND SECURITY
- A. How are computers used as home security systems? How do they work? How much do they cost? Where can they be found? Are they really effective?
- B. Government spying (espionage)--why do other governments want to steal our technology? How are they doing it?
- C. How are computers used by the military? Are strategic simulations really accurate? How are computers used in satellite communications? Armaments? Star Wars?

XI. THE COMPUTER AS AN AID TO MAN
- A. Computers in medicine--what are the different ways in which computers help patients and doctors? Will they ever replace doctors.
- B. Electronic mail and bullentin boards--what are they? How do they work? Who will use them? Are they any good? What impact will they have on our postal service?
- C. Computer-Aided Design (CAD)--how do architects and engineers use computers in their work?

Handout #2
Page 1 of 2
Computer Ethics and Impact

NAME _____
DATE _____
PERIOD _____

SUMMARY FORMAT

Author's Name: _____
Title of Article: _____
Name of Magazine/Book: _____
Date/Copyright Year: _____
Volume and/or Page number: _____

INTRODUCTION: (Main Ideas.)

_____

_____

_____

_____

_____

BODY:

1) _____

_____

2) _____

_____

3) _____

_____

4) _____

_____

5) _____

_____

CONCLUSION: Refer to the suggestions on the next page.

POSSIBLE SENTENCE STARTERS FOR CONCLUSION

I feel that.....because.....

I think that.....because.....

The main advantage of.....is.....

The main disadvantage of....is.....

We should. .....

I believe that.....

Computers are.....

From the article, I.....

In my opinion.....

I hope.....

I agree with the article because.....

I disagree with the article because.....

Probably in the future.....

Wouldn't it be terrific if.....

We shouldn't.....because.....

Our society will.....

Computers have tremendous impact on our society because.....

Computers will not have any impact on our society because.....

APPENDIX

TAXONOMY OF GOALS, OBJECTIVES AND STUDENT EXPECTATIONS
For Computer Science

GOALS:

1. The student will demonstrate competence in using computers.

2. The student wi use the computer as a tool for problem solving and decision making.

   The student will recognize the impact of computers in daily life.

4. The student will investigate educational and career opportunities in computer-related professions.

GOAL 1:

1. The student will demonstrate competence in using puters.

1.1. Interacts with prepackaged computer programs.

1.1.1. The student selects and uses appropriate resources (manuals, program, peripherals, etc.) for performing a task.

1.1.2. The studer adapts programs to solve specific problems.

1.1.3. The student evaluates and compares computer programs (prepackaged; and student's own).

1.2. Processes information according to a set of predefined computer rules: organized, coded, given meaning and transmitted.

1.2.1. The student demonstrates through a project the processing of information.

1.2.2. The student implements routines to process information through searching, sorting, deleting, updating, summarizing, storing, etc.

1.2.3. The student explains the major parts and functions of a computer system (e.g., CPU - registers, accumulators; memory - addressing; peripherals - cylinders, tracks, sectors).

1.3. Develops good programming style in a higher level language such as Pascal. Good programming style includes logical structure, documentation (readability), efficiency, elegance.

1.3.1. The student demonstrates the ability to clearly define problems and to subdivide a particular problem into logical subproblems.

1.3.2. The students designs structured solutions to problems - algorithms - by applying the principles of top-down design methodology

1.3.3. The student properly implements the available control structures - sequence, interation and branching - when coding algorithms into a specific high-level language.

1.3.4. The student designs and uses numeric and string arrays and matrices.

1.3.5. The student demonstrates the ability to anticipate, identify, isolate and correct errors.

1.3.6. The student enhances the readability and clarity of his or her program by including appropriate documentation.

GOAL 2.

2. The student will use the computer as a tool for problem solving and decision making.

2.1. Selects and uses appropriate data structures to solve problems.

2.1.1. The student describes the major types of data structures available to the high-level language being studied and understands their uses and limitations.

2.1.2. The student recognizes and appropriately utilizes elementary data structures in solving problems.

2.2. Creates and implements algorithms to solve problems.

2.2.1. The student recognizes and appropriately utilizes elementary algorithms in solving problems.

2.2.2. The student designs and implements his or her own algorithms in solving some types of programming problems.

2.3. Uses a computation/information system (computer or computer system) to solve challenging problems and make decisions.

2.3.1. The student creates and utilizes sequential data files for file-processing programs.

2.3.2. The student creates and utilizes random data files for file-processing programs.

2.3.3. The students easily uses mathematical and string manipulation functions specific to the high-level language being studied.

2.3.4. The student designs a variety of graphics programs.

2.3.5. The student experiences working as a team in a programming environment which simulates the actual field where each team is responsible for developing one module in a larger programming system.

2.4. Uses the computer for information storage and retrieval, simulation and modelling, quality or process control and decision making, computation, data processing.

2.4.1. The student recognizes and uses computer application tools.

2.4.2. The student values efficient information processing.

GOAL 3.

3. The student will recognize the impact of computers in daily life.

3.1. Recognizes ethical and social implications of computer use.

3.1.1. The student identifies computer applications in business, industry, scientific research, medicine, government, education, health and social services, recreation, creative arts, etc.

3.1.2. The student appreciates the economic benefits of computerization for society.

3.1.3. The student understands that computers can be used to effect distribution and use of economic and political power, and used in criminal and other anti-social activities that affect society in undesirable ways.

3.2. Demonstrates responsible use of computer systems.

3.2.1. The student accepts responsibility for following school and lab rules pertaining to computer ethics.

GOAL 4.

4. The student will investigate educational and career opportunities in computer-related professions.

4.1. Recognizes careers involving computers and the impact these careers have on society and the educational system.

4.1.1. The student identifies careers that involve computers directly (support, service, technical and scientific careers, data management, programming analysis, etc.)

4.1.2. The student compares educational requirements and opportunities for careers that involve computers.

SIGN-UP FOR COMPUTER TIME DURING WEEK: ____________________

TIME	MONDAY	TUESDAY	WEDNESDAY	THURSDAY	FRIDAY
	1. 2. 3. 4. 5.	1. 2. 3. 4. 5.	1. 2. 3. 4. 5.	1. 2. 3. 4. 5.	1. 2. 3. 4. 5.
	1. 2. 3. 4. 5.	1. 2. 3. 4. 5.	1. 2. 3. 4. 5.	1. 2. 3. 4. 5.	1. 2. 3. 4. 5.
	1. 2. 3. 4. 5.	1. 2. 3. 4. 5.	1. 2. 3. 4. 5.	1. 2. 3. 4. 5.	1. 2. 3. 4. 5.
	1. 2. 3. 4. 5.	1. 2. 3. 4. 5.	1. 2. 3. 4. 5.	1. 2. 3. 4. 5.	1. 2. 3. 4. 5.
	1. 2. 3. 4. 5.	1. 2. 3. 4. 5.	1. 2. 3. 4. 5.	1. 2. 3. 4. 5.	1. 2. 3. 4. 5.
	1. 2. 3. 4. 5.	1. 2. 3. 4. 5.	1. 2. 3. 4. 5.	1. 2. 3. 4. 5.	1. 2. 3. 4. 5.
	1. 2. 3. 4. 5.	1. 2. 3. 4. 5.	1. 2. 3. 4. 5.	1. 2. 3. 4. 5.	1. 2. 3. 4. 5.

NAME _____

DATE _____

PERIOD _____

PROGRAM _____

GRADING SHEET

_____

te due: _____

_____

da_ received: _____

_____

extra points: _____

_____

_____

flowchart: _____

_____

documentation: _____

_____

translation to BASIC: _____

_____

output validation: _____

_____

other: _____

_____

points for program: _____

plus extra points: _____

FINAL SCORE: _____

COMME_ :

A Motivation Technique in Computer Programming*

Theodore Kosko, a high school computer science teacher in Little Falls, New Jersey, recommends a technique he finds conducive to motivating students to excel in his computer programming classes. He establishes a fictitious software company where each student is to be employed as a Junior Programmer. Based upon the merit of his or her work, each student can earn a promotion to a more responsible position. Using current classified ads, he determines the salaries for each position and creates a chart such as the one below.

Position	Employee	Salary
Information Systems Director		$75,000
Programming Mgr.		$60,000
Project Leader		$52,000
Systems Programmer		$47,000
Senior Analyst		$43,000
Systems Analyst		$33,000
Programmer		$25,000
Junior Programmer		$20,000
. .		
Programming Instructor		$1.25/hr.

Each position has a particular number of slots. After each assignment is graded, a new promotion chart, based upon these marks, is posted on the bulletin board, showing each student's name in the appropriate position. Mr. Kosko allows the authors of the seven best programs for the first assignment to move up to Programmer. The next promotion allows a maximum of six to move up to Systems Analyst, leaving room for six more people to advance from Junior Programmer to Programmer. The following round advances five to Senior Analyst, then four to Systems Programmer, then three to Project Leader, then two to Programming Manager, and finally one to Information Systems Director!

Test or quiz results, class participation and homework quality are used to break ties in assignment grades. Students advance only on the basis of quality work, and only a limited number of higher positions are available - just like in the real business world.

Mr. Kosko has found students eager to work harder so they can advance in position. All students' work improves with this system, as they become more conscious of structured methodology, program readability, good documentation and accurate output. And curiosity about the responsibilities of each job position encourages students to seek more information concerning computer careers. Good working

relations with others is emphasized so that a healthy and fun competition develops within the class that promotes continued cooperation as well among the students.

*"How To Motivate Your Computer Programming Classes," Theodore Kusko, The Computing Teacher, February, 1985, page 33.

ASCII CHARACTER CODES

The Apple II series computers store each character as a numeric code using the American Standard Code for Information Interchange (ASCII). Codes 32-127 are the ASCII code numbers that are used for characters. (Codes 0-31, used for various control functions are not shown here.)

To display the code number for a character, use the ASC function. To display the character that corresponds to a code number, use the CHR$ function.

Examples:

```
PRINT ASC("R")
82
```

```
PRINT CHR$(82)
R
```

ASCII Code	Character	ASCII Code	Character	ASCII Code	Character	ASCII Code	Character
32		56	8	80	P	104	h
33	!	57	9	81	Q	105	i
34	"	58	:	82	R	106	j
35	#	59	;	83	S	107	k
36	$	60		84	T	108	l
37	%	61	=	85	U	109	m
38	&	62		86	V	110	n
39	'	63	?	87	W	111	o
40	(	64	@	88	X	112	p
41	)	65	A	89	Y	113	q
42	*	66	B	90	Z	114	r
43	+	67	C	91	[	115	s
44	,	68	D	92	\	116	t
45	-	69	E	93	]	117	u
46	.	70	F	94	^	118	v
47	/	71	G	95	_	119	w
48	0	72	H	96	`	120	x
49	1	73	I	97	a	121	y
50	2	74	J	98	b	122	z
51	3	75	K	99	c	123	
52	4	76	L	100	d	124	
53	5	77	M	101	e	125	
54	6	78	N	102	f	126	
55	7	79	O	103	g	127	

The following ASCII codes represent characters that either are difficult to see with some monitors or printers or else are not visible characters at all: 32 (space); 39 (apostrophe); 44 (comma); 45 (hyphen); 46 (period); 94 (caret); 95 (underscore); 96 (grave accent); 126 (tilde); 127 (deletion symbol).

RESOURCES

RECOMMENDED TEXTBOOKS

(Listed within parentheses are schools who have used or are currently using the textbook.)

A Guide to Programming in Applesoft. Presley. Von Nostrand Reinhold Company. 1984.

Advanced Structured BASIC. Clark and LaBarre. South-Western Publishing Company. 1985.

Apple BASIC. Haskell. Prentice-Hall. 1982.

Apple Files in BASIC. Miller. Reston Publishing Company. 1982.

BASIC - A First Course. Thompson. Charles E. Merrill. 1981.

BASIC - A Structured Approach. Kittner and Northcutt. Benjamin/Cummings. 1985.

Basic BASIC Programming. Peluso, Bauer and DeBruzzi. Addison-Wesley. 1981.

BASIC For Students: With Applications. Trombetta. Addison-Wesley 1981.

BASIC Programming for the Classroom Teacher. Miller, Chaya and Santora. Teachers College Press. 1982.

BASIC Programming Using Structured Modules. Barron. CBS Publishing. 1983.

Computer Fundamentals with BASIC Programming. Mandell and Mandell. West Publishing Company. 1985.

Computer Programming in BASIC. Myers, Elswick, Hopfensperger and Pavlovich. Houghton Mifflin. 1986.

Computer Programming in the BASIC Language. Golden. Harcourt Brace Jovanovich. 1980.
(Schools: Farrington, Kaiser, Moanalua, Nanakuli, Roosevelt, Waipahu)

Computers and Data Processing - Concepts and Applications with BASIC. Mandell. West Publishing Company. 1982.

Data File Programming in BASIC - A Self-Teaching Guide. Finkel and Brown. John Wiley and Sons, Inc. 1981.

Data Processing - An Introduction with BASIC. Spencer. Charles E. Merrill Publishing Company. 1982.

Data Structures for Personal Computers. Langsam, Augenstein and Tenenbaum. Prentice-Hall, Inc. 1985.

Fundamentals of BASIC Programming - A Structured Approach. Mandell and Mandell. West Publishing Company. 1985.

Introduction to BASIC Programming. Shell and Cashman. Anaheim Publishing Company. 1982.
(Schools: McKinley, Kailua)

Introduction to Programming in Basic ... le II/IIe/IIc. Bitter. Random House Courseware. 1985.

More TRS-80 BASIC: A Self-Teaching ..., Zamora and Albrecht. John Wiley & Sons, Inc. 1981.
(Schools: Moanalua)

Practicing Programming on the Apple II/IIe. Marrapodi, Budin and Guzman. Random House, Inc. 1984.

Practicing Programming on the TRS-80. Marrapodi, Budin and Guzman. Random House, Inc. 1984.

Problem Solving and Structured Programming in BASIC. Koffman and Friedman. Addison-Wesley. 1979.

Problem Solving with BASIC. Dillman. Holt, Rinehart and Winston. 1983.

Programming in Apple BASIC. Dielsi, Grossman and Tucciarone. CBS College Publishing. 1984.

Programming in BASIC. Cum...s and Kuech...n. Charles Merrill Publishing Company. 1983.
(Schools: Pearl City, Waialua

Programming in BASIC. McRitchie. Holt, Rinehart and Winston. 1982.
(Schools: Farrington, Kalani, Waipahu)

Structured BASIC. Clark and Drum. Southwestern Publishing Company. 1983.
(Schools: Kaimuki)

The Basics of BASIC. Gomez. Holt, Rinehart and Winston. 1983.

The Mind Tool. Graham. West Publishing Company. 1983.
(Schools: Kaiser)

Using Computers. Elgarten, Posamentier and Moresh. Addison-Wesley. 1984.
(Schools: Castle, Waianae)

RECOMMENDED TEACHING AIDS AND REFERENCES
(TUTORIALS, ACTIVITY SHEETS, WORKBOOKS)

GENERAL (Applicable to most major microcomputers)

BOOKS (WORKBOOKS AND REFERENCE BOOKS)

BASIC Discoveries. Malone & Johnson. Creative Publications. 1982.

Duplicating Masters - Experiencing BASIC. Mulcahy. Media Materials, Inc. 1984.

Everybody's BASIC. Richardson. Meka Publishing Company. 1983.

Experiencing BASIC - Task Cards. Mulcahy. Media Materials, Inc. 1984.

Fifty BASIC Exercises. Lamoitier. Sybex. 1980.

Graphics Discoveries, Book I and II. Johnson. Creative Publications 1984.

Hands-On BASIC. Peckham. McGraw-Hill Publishing Company. 1984. (specify computer)

How to Build a Program. Emmerichs. Prentice Hall Press. 1983.

Micros 1-10. Ruby and Roberts. J. Weston Walsh. 1984.

Problems for Computer Solution. Rogers. Creative Computing Press. 1979.

Problems for Computer Solutions Using BASIC. Walker. Winthrop Publishers, Inc. 1980.

Problems Solving with the Computer. Henley & Henley. Media Materials Inc. 1983.

Problems for BASIC Discoveries. Johnson & Malone. Creative Publications. 1984.

Structured Design and Programming. MECC. 1984.

Teaching BASIC: Thirty Lesson Plans, Activities and Quizzes - Volume II, TRS-80 Models III, IV. Erickson & Vonk. Learning Publications, Inc. 1983.

The BASIC Workbook. Schoman. Hayden. 1977.

GENERAL

SOFTWARE

Global Program Line Editor; Beagle Brother.

Practicing Programming; Random House

Visible Bubble Sort; Softdisk Magazine

OTHER CLASSROOM AIDS

MECC Computer Parts Kit; MECC.

Quick Reference Guide (specify the computer); John Wiley & Sons.

The BASIC Conversions Handbook for Apple, TRS-80 and PET Users. Brain, et. al. Hayden. 1982.

APPLE

BOOKS (WORKBOOK AND REFERENCE BOOKS)

A Guide to Programming in Applesoft. Presley. Von Nostrand Reinhold Company. 1984.

Apple Basic: Data File Programming. Finkel & Brown. Creative Publications. 1982

Apple BASIC for Business - for the Apple II. Parker and Stewart. Reston Publishing Company, Inc. 1981.

Apple Graphics Activities Handbook. Creative Publications. 1984.

Graphics Cookbook for the Apple. Wadsworth. Hayden.

Graphics Discoveries Books I and II. Johnson. Creative Publications.

Practicing Programming on the Apple II/IIe. Marrapodi, Budin, Guzman. Random House, Inc. 1984.

Programming Special Projects. MECC. 1984.

Spotlight on Basic for the Apple II/IIe/IIc. Feist and Rohman. Random House, Inc. 1985.

The Apple Program Factory. Stewart. McGraw Hill 1985.

The BASIC Handbook. Lien. Compusoft Publishing. 1985.

APPLE

SOFTWARE

BASIC Tutor Series: Advanced Topics; Educational Courseware

BASIC Tutor Series: Creating Music and Sounds; Educational Courseware

BASIC Tutor Series: Graphics Commands; Educational Courseware

BASIC Tutor Series: Shapes and Pictures; Educational Courseware

BASIC Tutor Series: Text File Commands; Educational Courseware

BASIC Tutor Series: Set; Educational Courseware

Computer Graphics; Innovative Programming Associates

Hands on BASIC Programming; Peachtree Software, Inc.

Hi-Res Secrets Graphics Applications Systems; Avant-Garde

How to Program in Applesoft Basic; Hayden Software

Let's Explore BASIC; Media Materials, Inc.

Programming Animation & Graphics for the Apple; Media Materials, Inc.

Scott, Foresman Computer Literacy Courseware Series; Scott, Foresman and Company

The Basics of BASIC; Focus Media

Visible Bubble Sort Program; Softdisk Magazine, P. O. Box 30008 Shreveport, LA 71130-0008; (Program written by Thomas G. Pink)

IBM PC

BOOKS (WORKBOOKS AND REFERENCE BOOKS)

BASIC Exercises for the IBM-PC. Lamoitier. Sybex. 1982.

Graphics Programming on the IBM Personal Computer. Volkstorf. Prentice-Hall, Inc. 1984.

Hands-On BASIC for the IBM-PC. Peckham. McGraw-Hill Book Company. 1983.

Programming Special Projects. MECC. 1984.

Programming the IBM Personal Computer: BASIC. Graham. Holt, Rinehart and Winston. 1982.

SOFTWARE

Beyond Basic BASIC; Personally Developed Software

The Basics of BASIC; Focus Media

TRS-80

BOOKS (WORKBOOKS AND REFERENCE BOOKS)

A Guide to Programming in Level II BASIC-TRS-80. Presley. Von Nostrand Reinhold Company. 1982.

Advanced BASIC. Radio Shack. 1982.

BASIC Disk I/O Faster & Better. Rosenfelder. I.J.G. 1982.

Business Programming Applications. Barden. Radio Shack. 1982.

Structured Program Design with TRS-80 BASIC. Dwyer & Critchfield. McGraw Hill Book Company. 1984.

TRS-80 Graphics. Grillo & Robertson. William C. Brown Company. 1981.

TRS-80 Level II BASIC. Albrecht, Inman, & Zamora. John Wiley & Sons, Inc. 1980.

RECOMMENDED PERIODICALS

GENERAL

BYTE; 70 Main Street, Peterborough, NH 03458

Classroom Computer Learning; 5615 W. Cermak Road, Cicero, IL., 60650

Education Computer News; 1300 N. 17th Street, Arlington VA 22209

Electronic Learning; Scholastic Inc., P.O. Box 645, Lyndhurst, NJ 07071-9986

InfoWorld; P.O. Box 1019, Southeastern, PA 19398-9981

Personal Computing; P.O. Box 2941, Boulder, Co. 80321

Teaching and Computers; Scholastic Inc., P.O. Box 645, Lyndhurst, NJ 07071-9986

T. H. E. Journal; P.O. Box 15126, Santa Ana, CA 92705-0126

The Computing Teacher; University of Oregon, 1787 Agate Street, Eugene, OR 97403-1923

PERIODICALS FOR APPLE COMPUTER

A+; P.O. Box 2964, Boulder, CO 80321

Apple Education News; 20525 Mariani Avenue, Cupertino, CA 95014

Incider; P.O. Box 911, Farmingdale, NY 11737

Nibble; P.O. Box 325, Lincoln, MA 01773

PERIODICALS FOR IBM PC

PC - The Independent Guide; 1 Park Avenue - 4th Floor, New York, NY 10016

PC WEEK; 1 Park Avenue - 4th Floor, New York, NY 10016

PC WORLD; P.O. Box 6700, Bergenfield, NJ 07621

PERIODICALS FOR TRS-80 COMPUTER

80 Micro; CW Communications, 80 Pine Street, Peterborough, NH 03458

AUDIOVISUAL SERVICES
641 18th Avenue
Honolulu, HI 96816

Ph: 732-2824

16mm Films

7705 AND WHAT OF THE FUTURE?
Films Incorporated, 1981
40 min.; J-H*
Will the recent developments in electronic microcircuitry result in a better or worse life for the average person? Will people lose jobs once thought secure? Will the technology be used to replace people on monotonous or dangerous jobs? Visits to the Washington D.C. Metro (subway), a Dallas supermarket, and a Scottish hospital illustrate the benefits and problems.

7724 THE COMPUTER AND YOU - AN INTRODUCTION
Handel Film Corporation, 1983
16 min.; E-J
A primer for computer operations designed for audiences who have no prior knowledge in this field. The computer terms come to life by watching a student developing a program about the states in the USA and the provinces of Canada.

6702 COMPUTER COLOR GENERATIONS
United States Department of Energy, 1972
23 min.; J-H C
Discusses new techniques in computer technology which virtually eliminate the extra cost of color in computer displays. Includes research on thermonuclear problems, lasers, engineering and three dimensional problems.

6703 COMPUTER FLUID DYNAMICS
United States Department of Energy, 1969
24 min.; J-H C
Demonstrates the power of today's giant electronic computers for solving problems that previously were impractical to undertake. Presents a wide range of fluid flow problems, shows several examples of fluid flow calculations, and describes how computer calculations are accomplished.

7940 COMPUTER: TOOL FOR THE FUTURE
National Geographic, 1984
23 min.; J-H T
The film begins with the human need to compute, surveying several computing devices that preceded the chip and focusing on significant computer applications. Computer careers are considered--everything from the military to music. This film stresses the importance of computers in our modern society.

7682 COMPUTERS AND THE FUTURE
Time-Life Media, 1982
30 min.; J-H
Combining documentary techniques with vignettes, the film explores our growing relationship with communications technologies such as interactive computers, cable television and video discs. The program explores the effect of this new media form on the way we live, work and play. Futurologist Peter Schwartz is host.

7798 COMPUTERS: THE FRIENDLY INVASION
Walt Disney Educational Media Company, 1982
20 min.; E-H
Computer graphics and scenes from the Disney feature "Tron" illustrate some computer applications in an entertaining film that introduces students to a future resource. They are introduced to how computers work, the many tasks they can perform, and the opportunities they offer in science and the arts.

7668 COMPUTERS: TOOLS FOR PEOPLE
Churchill Films, 1983
22 min.; E-H T
Shows how computers are used in many ways: for file management; control of other machines; support of creative work; and for mathematical tasks including modeling. Demonstrates how applications are developed through research, flowcharting, programming and debugging. Emphasizes the human responsibility for computer performance and the excitement of people creating their own tools.

7922 DON'T BOTHER ME, I'M LEARNING:
ADVENTURES IN COMPUTER EDUCATION:
MCHT, 1981
24 min.; E-H T
This motivating film demonstrates uses of a computer in a classroom. Teachers, parents and students all eagerly discuss the vast uses of the computer.

7473 MIND MACHINES, THE PARTS I & II
Time-Life Media, 1979
57 min.; H C
The controversy surrounding artificial intelligence is examined. Computer fundamentals are explained and compared to human intelligence. Limitation of computers to memory and calculations functions are used to argue the nature of human intelligence which includes judgement, common sense, etc.

7707 NOW THE CHIPS ARE DOWN, PARTS I & II
Films Incorporated, 1981
50 min.; J-H C A T
Microprocessors smaller than a postage stamp have the power

of room-sized computers of a generation ago. We hear a machine that can read aloud, see a driverless tractor and a warehouse that needs no staff among the samples of the wonders created by cheap computer power. We also learn how microcomputers are made, and hear predictions of the future changes.

7929 ROBOT REVOLUTION, THE
EBEC, 1984
19 min.; J-H C
This probing look at robots--their capabilities and their limitations--explores their potential for improving the quality of life and their threat to the labor force as they enter the workplace. Shows the effects of using computers and robots in medicine, research, business and industry.

7591 ROBOTS - INTELLIGENT MACHINES SERVING MANKIND
Pacific Resources, Inc., 1981
14 min.; J-H
The film presents a report of an army of "intelligent machines" taking over more and more jobs that were previously performed by workers. Benefits derived by this major revolution in computer technology are: safer working conditions, high quality products and more efficient use of resources. This report includes robots with limited touch, sight and judgement - first steps in the mechanical evolution.

6858 TIC - INDEX TO ENERGY
United States Department of Energy, 1977
6 min.; H C A
Describes the Technical Information Center (TIC) of the Department of Energy at Oak Ridge, Tennessee. The computerized facility gathers, abstracts and catalogues technical reports and published scientific papers from sources around the world. This material is evaluated and part of it becomes a permanent part of the data bank of technical and scientific energy information.

7901 WELCOME TO THE FUTURE: COMPUTERS IN THE CLASSROOM:
FI, 1982
28 min.; C T
In plain language, this film introduces teachers to computer literacy: programming languages, software and the variety of ways computers can be used in schools, such as computer assisted instruction. This film helps demystify computers and shows how teachers and students can become friends with a machine.

*Codes for grade or ability levels are:
E = elementary; J = junior high; H = high school; C = college;
A = adult; T = teacher.

TECHNICAL ASSISTANCE CENTER
3645 Waialae Avenue, Room B-6
Honolulu, Hi 96816

Ph: 735-2825

Videotapes

R199-1 BEYOND THE PROGRAM
Great Plains National, 1980
20 mins., Color (Business Computing...Cut Down to Size); A*
Outlines elements of data reliability and accuracy and stresses the need for safeguards. LOAN ONLY. NOT FOR CATV USE. NOT AVAILABLE TO PUBLIC LIBRARIES.

R197-1 COMMUNICATING WITH YOUR COMPUTER
Great Plains National, 1980
27 min., Color (Business Computing...Cut Down to Size); A
Introduces and compares elements of programming languages. LOAN ONLY. NOT FOR CATV USE. NOT AVAILABLE TO PUBLIC LIBRARIES.

0890-1 COMPUTER COLOR GENERATIONS
ERDA, 1972
23 mins., Color; J-H C
Discusses new techniques in computer technology which virtually eliminates the extra cost of color in computer displays. Includes research on thermonuclear problems, laser, engineering and three dimensional problems.

0911-1 COMPUTER FLUID DYNAMICS
ERDA, 1969
24 mins., Color; J-H C
Demonstrates the power of today's giant electronic computers for solving problems that previously were impractical to undertake. Presents the wide range of fluid flow calculations and describes how computer calculations are accomplished.

0815-2 COMPUTER FRIEND
WPBT Public Television, 1976
30 mins., Color (Que Pasa, USA?); H A
Carmen fills out an application for a computer program that matches up people of similar interests and family backgrounds. Spanish/English program.

1768-1 COMPUTER SHOW #1, THE
Oceanic Cablevision, Inc., 1984
27 min., Color (The Computer Show); J-H A
In a magazine format: computer applications in the travel industry; tutorial on the components functions and operation of a microcomputer; care and maintenance of computers; introduction to programming in Logo.

1768-2 COMPUTER SHOW #2, THE
Oceanic Cablevision, Inc., 1984
30 min., Color (The Computer Show); J-H A
Continuation of show #1 with emphasis on the care and maintenance of computers, use of word processors and printers.

1804-1 COMPUTER SHOW #3, THE
Oceanic Cablevision, Inc., 1984
30 min., Color (The Computer Show); J-H A
David Kobashigawa of Radio Shack demonstrates the use of a computer spreadsheet. The film defines some computer languages, gives computer care tips and describes various types of printer paper.

1854-1 COMPUTER SHOW #4, THE
Oceanic Cablevision, Inc., 1984
30 min., Color (The Computer Show); J-H A
Demonstrates computer programming in BASIC and explains some of its terms. Describes what computers can do and how they work. Explores the serious problem of software piracy.

1854-2 COMPUTER SHOW #5, THE
Oceanic Cablevision, Inc., 1984
30 min., Color (The Computer Show); J-H A
Briefly demonstrates features of a computer operating system designed for multi-user business applications, the Northstar "Dimension" system; this is followed by a demonstration of software called "Color Paint." Both programs are designed for IBM PC computers.

1851-2 COMPUTER SHOW #6, THE
Oceanic Cablevision, Inc., 1984
30 min., Color (The Computer Show); J-H A
Minidocumentaries in this program feature computers: computerized music, computers in designing and manufacturing, a young science fair winner who is a computer whiz, a new way of notating dance, and the work of robots, present and future.

1282-1 COMPUTERS
Hawai'i Public Television, 1980
60 mins., Color (Dialog); H C A
Presents a group of computer experts who discusses the use of computers, their advantages and possible disadvantages. The question is--what is the future of computers, will they compete with people for jobs? NOT FOR CATV USE.

1278-4 COMPUTERS
Hawai'i Public Television, 1980
10 mins., Color (Dialog); H C A
Edited version of "Computers" without the panel discussion. Shows only the mini-documentary of the topic up for discussion.

0112-1 COMPUTERS
Hawai'i Public Television, 1982
59 Mins., Color (Dialog); H C A T
This program examines the numerous functions and disadvantages of having personal computers. It also describes the different brands of computers and their most effective use, especially by the average person. NOT FOR CATV USE.

0088-2 COMPUTERS
Hawai'i Public Television, 1982
6 min., Color (Dialog); H C A T
Edited version of "Computers" without the panel discussion. Shows only the mini-documentary of the topic up for discussion.

1756-1 COMPUTERS
WETA-TV, Washington, D.C., 1983
26 min., Color (Spaces); J-H
Minidocumentaries in this program feature computers: computerized music, computers in designing and manufacturing, a young science fair winner who is a computer whiz, a new way of notating dance, and the work of robots, present and future.

R129-1 COMPUTERS AND THE FUTURE
Time-Life Video, 1982
30 mins., Color; J-H A
Combining documentary techniques with vignettes, the film explores our growing relationship with communications technologies such as interactive computers, cable television and media forms on the way we live, work and play. Futurologist Peter Schwartz is host. LOAN ONLY. NOT FOR CATV USE. NOT AVAILABLE TO PUBLIC LIBRARIES.

DATABASE PROCESSING SYSTEMS (COMPUTERS AT WORK SERIES-#10)
BNA Communications, Inc. 1980
30 mins, Color; H A
A banking application serves as a case study to define the characteristics of database processing and show how it differs from file processing. A limited number is available. If interested, call Computer Education, Office of Instructional Services, at 395-8916.

1664-2 DATA PROCESSING
Kapi'olani Community College, 1983
12 mins., Color (A Career in Focus); J-H
In the business world, computers play a major role in data processing, accounting, and record keeping. This program describes the duties and work of computer operators, computer programmers, data entry clerks and control clerks. KCC offers a two-year course in computer education.

L001-1 EVOLUTION: COMPUTERS, YESTERDAY AND TODAY (COMPUTERS AT WORK SERIES-#1)
BNA Communications, Inc., 1983
30 min.; J-H A
This film presents the history of four generations of computers. Included are the people and the companies that developed them.

L015-1 FILE STRUCTURES - BASIC POWERS PROGRAM 8
University of California, EMC, 1982
19 mins., Color; H A
Continues the discussion of input and output, but with emphasis on external storage of programs and data on disk and tape units. Concludes by examining several popular microcomputer operating systems.

R196-1 FITTING OUT
Great Plains National, 1980
15 mins., Color (Business Computing..Cut Down to Size); A
Offers guidelines on determining the capabilities of computer systems (size, storage maintenance). LOAN ONLY. NOT FOR CATV USE. NOT AVAILABLE TO PUBLIC LIBRARIES.

L013-1 HARDWARE & SOFTWARE (COMPUTERS AT WORK SERIES-#3)
BNA Communications, Inc., 1980
30 mins., Color; J-H A
In-depth look at hardware and software, memory and data storage, programs, languages and operating systems. Highlighted by a visit to a modern computer cent r.

R195-1 MEASURING UP
Great Plains National, 1980
15 mins., Color (Business Computing..Cut Down to Size); A
Details applications and types of small computers and their integration into a business. LOAN ONLY. NOT FOR CATV USE. NOT AVAILABLE TO PUBLIC LIBRARIES.

L014-1 SUBSCRIPTED VARIABLES AND ARRAYS - BASIC POWERS PROGRAM 5
University of California, EMC, 1982
22 mins., Color; H A
Introduces the use of dimensioned variables, which contain more than one set of values or groups of characters. Concludes by showing how to design a useful "bubble-sort" routine.

1593-1 TIC - Index to Energy
U.S. Department of Energy, 1977
6 min.; Color; C A
Describes the Technical Information Center (TIC) of the Department of Energy at Oak Ridge, Tennessee. The computerized facility gathers, abstracts and catalogues technical reports around the world. This material is evaluated and becomes a permanent part of a data bank of technical and scientific energy information.

R198-1 UNDERSTANDING SOFTWARE
Great Plains National, 1980
16 mins., Color (Business Computing...Cut Down to Size); A
Discusses types, applications and the choosing of software.
LOAN ONLY. NOT FOR CATV USE. NOT AVAILABLE TO PUBLIC
LIBRARIES.

1745-1 WHY IN THE WORLD #245
WNET & Satellite Education Services, Inc., 1984
30 min., Color (Why in the World); J H A
Topic: Computers and the changes they bring to
America---how people live and work. Guest: John F. Akers,
President of IBM Corporation.

*Codes for grade or ability levels are:
E = elementary; J = junior high; H = high school; C = college;
A = adult; T = teacher.