ED 264 850 IR 011 937

AUTHOR	Lopez, Antonio M., Jr.
TITLE	Computer Literacy Training Modules for Special Educators. Bulletin 1749.
INSTITUTION	Louisiana State Dept. of Education, Baton Rouge.
SPONS AGENCY	Office of Special Education and Rehabilitative Services (ED), Washington, DC.
PUB DATE	85
GRANT	G008400622
NOTE	37p.
PUB TYPE	Guides - Classroom Use - Guides (For Teachers) (052)
EDRS PRICE	MF01/PC02 Plus Postage.
DESCRIPTORS	Computer Assisted Instruction; *Computer Literacy; Computer Software; *Curriculum Development; Elementary Secondary Education; Instructional Innovation; Learning Modules; *Microcomputers; Programing; *Programing Languages; Resource Materials; *Special Education; State Programs; Teaching Methods
IDENTIFIERS	*BASIC Programing Language; *Computer Uses in Education; Louisiana

ABSTRACT

Designed to provide Louisiana special educators with the background information they need to develop a computer literacy curriculum for use with their students, this document is divided into four training modules: (1) Introduction to Computer Literacy and Computer History; (2) Introduction to Computer Hardware and Software; (3) Introduction to Computer Programming with BASIC; and (4) Evaluation of Application Software. To assist teachers in identifying their skills development, lists of objectives and related activities introduce each module. Modules 1 and 2 present information on the concept of computer literacy, the history of computers and computer use, computer hardware and software, computer assisted instruction (CAI), and programming languages. Module 3 provides step-by-step instructions for writing a computer program in BASIC and addresses the nature of computer programing as a problem-solving procedure. Module 4 presents suggestions for evaluating computer software and identifies three basic stages in an evaluation: the classification, description, and observation stages. Finally, a chart is included which lists the characteristics of five types of CAI software--drill and practice, tutorial, testing, simulation, and dialog. (JB)

1R

ED264850

Computer Literacy Training Modules for Special Educators

Developed by
Antonio M. Lopez, Jr., Ph.D.

STATE OF LOUISIANA
UNION · JUSTICE · CONFIDENCE

1985
Bulletin 1749

Louisiana State Department of Education
Thomas G. Clausen, State Superintendent

Office of Special Educational Services
Irene M. Newby, Assistant Superintendent

EC011937

FOREWORD

The Computer Literacy Training Modules have been developed under a federal training grant to ensure that every effort is made to provide Louisiana's exceptional students with the educational advantages of the technological revolution.

The advent of the microcomputer in the classroom has proven to be a challenge for educators. The modern technology of the microcomputer has added many options for instruction and for new instructional materials. Teachers and administrators are now faced with many decisicns concerning such crucial areas as long-range planning, selection and evaluation of educational software, and curricular development.

We, as educators, will need to identify specific computer literacy skills that will be useful to us as we work toward addressing these concerns for successful integration and appplication of computer technology to the instruction and administration of exceptional students. It is our hope that these training modules will be a valuable tool to special educators.

Sincerely,

Thomas G. Clausen

Thomas G. Clausen, Ph.D.

ACKNOWLEDGMENT

The development of the Computer Literacy Training Modules was supported by PL 94-142, Handicapped Training Grant, No. G008400622, (Project No. 029HH40009), Part D, from the Office of Placement and Assistance Management, Office of Special Educational Services and Rehabilitative Services, U.S. Department of Education. However the opinions expressed herein do not necessarily reflect the opinion or policy of the U.S. Department of Education, and no offical endorsement by the U.S. Department of Education should be inferred.

Computer Literacy Training Modules

Antonio M. Lopez, Jr., Ph.D., Author
Mathematical Sciences Department
Loyola University
New Orleans, Louisiana

State Department of Education Advisory Committee

Mr. Donald E. Butcher, Project Director
Louisiana Learning Resources System

Mr. Steve Vidrine, LANSER Coordinator
Office of Special Educational Services

Dr. Catherine Nelson, Bureau Director
Bureau of Interagency Coordination

Dr. Charlene M. Bishop, Director
Louisiana Learning Resources System

Ms. Shirley McCandless, Administrative Officer
Management Information Services

State Advisory Committee

Dr. Dan Trimble
Supervisor of Special Education
Madison Parish Schools
Tallulah, LA

Mr. Leo Stanford, Supervisor
Pupil Appraisal Services
Calcasieu Parish Schools
Lake Charles, LA

Dr. Shirley T. Becnel, Principal
Westgate School
Metairie, LA

Ms. Evelyn Sisco, Teacher
Pineville High School
Pineville, LA

Ms. Sharon Bell
Educational Program Researcher
Orleans Parish Schools
New Orleans, LA

Mr. Warren Figueiredo, Teacher
Louisiana School for the Visually
Impaired
Baton Rouge, LA

Dr. William F. Grady, Professor
Educational Technology
Adminstration and Foundations
Louisiana State University
Baton Rouge, LA

Dr. Dorothy Judd, Assistant Professor
Department of Education
Southeastern Louisiana University
Hammond, LA

Mrs. Patricia Faser
Associate Professor
Computer Science Department
Southern University
Baton Rouge, LA

Mrs. Lou Price, Principal
Northwest State School
Bossier City, LA

TABLE OF CONTENTS

MODULE 1
Introduction to Computer Literacy and Computer History

Objectives

To define computer literacy and discuss the various interpretations of the definition.

To demonstrate an awareness of the computer's impact on society by listing:
1. Three devices in the home that contain computers.
2. Five kinds of businesses that use computers and briefly explaining how each one uses the computer.
3. Three governmental agencies that use computers and briefly explaining how each one uses the computer.

To demonstrate knowledge of the pre-computer era by:
1. Identifying at least three early calculating devices.
2. Associating names with contributions of at least three major figures.

To demonstrate knowledge of the development of the computer by:
1. Listing the major characteristics of each of the five generations of computers.
2. Associating names with contributions of at least three major figures.

Activities

1. Give examples of computers used in numerous organizations to keep track of information and to control other machines.
2. Discuss the statement, "Local, state, and Federal governments could not function without computers."
3. State at least one way each of the following government agencies uses computers: IRS, DOD, NASA, FBI, and USPS.
4. Discuss how computers could be used in education and medicine.
5. Show pictures of early calculating devices and discuss their impact on society.
6. Demostrate the use of a slide rule.
7. Show pictures of vacuum tubes, transistors, and integrated circuits and discuss the impact of each technology on the computer.
8. Compare and contrast the computer gernerations in physical sizes of equipment, electrical energy used, calculating speed, and cost.
9. List the names of the computer manufacturers that you know.
10. List those abilities that you would like future computers (those beyond the Fifth generation) to have.

Introduction to Computer Literacy and Computer History

Computer literacy is not a static concept, but a dynamic one. As technology changes the world in which you live, so does your level of understanding and appreciation for that technology change. The purpose of this module is the establishment of a foundation for future growth in computer literacy. The Louisiana Task Force on Computer Literacy in its "Position Paper on Computer Literacy" (July 1983) defined computer literacy as the knowledge of the capabilities, limitations, applications, and implications of computer technology. Furthermore, the Task Force stated that a computer-literate person is one who understands what a computer is, who can use it to solve some problems, who can converse in computer-related terms, and who understands the impact of the computer on jobs and society. In fall 1984, a Computer Literacy Curriculum Guide Writing Committee was formed in response to the decision by the Louisiana State Board of Elementary and Secondary Education that freshmen entering high school in 1985-86 must have one-half credit in Computer Literacy as a graduation requirement. This guide, Bulletin 1739, was piloted during spring 1985 and revised during summer 1985 for final publication. Teachers of computer literacy should familiarize themselves with this guide before proceeding further with this training module.

Presently, there are two major schools of thought involving what should or should not be in a computer literacy course. The "bone of contention" involves the role of programming. At one extreme there is the group that would not have programming involved in a computer literacy course at all. While at the other extreme there is the group that would have programming playing a major part (50% or more) of the computer literacy course. Analogies abound but the first group usually uses the Auto Mechanic Model to explain its position; that is, in order to drive a car you do not have to be an auto mechanic. Hence, you are to infer that in order to be computer literate you do not have to be involved with programming the computer, just able to use existing programs. The second group usually uses the English Literacy Model as its analogy; that is, in order to be literate one must be able to read and write English. Hence, you are to infer that in order to be computer literate you must be able to read and write a programming language. The approach that will be taken in these training modules will be one of compromise between the two extremes. This is the position that has been suggested by Bulletin 1739.

A change in technology always has a large impact on society. Consider, if you will, the social impact of the Industrial Revolution, the explosion of the Nuclear Bomb, and the flight of the Space Shuttle. The Computer Revolution is no different. It began in the late 40's due in part to a change in technology, using a vacuum tube to store on/off information. Since that time developments in technology have come about very quickly. One expert in the computer industry said that if the auto industry had grown as quickly as the the computer industry had grown in the last 30 years, a Rolls Royce would cost $2.50 and get 2 million miles to the gallon of gasoline. You can probably get a better grasp of the impact of the Computer Revolution if you can imagine today's world with no computers. There would be no digital clocks to wake you up in the morning; no microwave ovens to cook your breakfast; no transportation capabilities like jet airplanes and rail systems like BART; even some of our cars would not function because they have computer-controlled ignition systems. There would be no telephone system as we know it; getting a check cashed at a bank would be out of the question, and most of the electrical plants would have to be closed because they would be too dangerous to operate. This list goes on and on, and it can make many days of interesting discussions in any classroom.

All of these facts notwithstanding, you must look for the roots of the revolution not in the technology but in the goals that society has set for itself. To quote Benjamin Franklin, "Necessity is the mother of invention." Since the time of the ancients, society has been interested in counting things. By counting how many sheep you had, how many baskets of wheat were in your barn, and so forth you could determine how wealthy you were. The abacus is probably the oldest instrument known to have been used for counting, and there is a path of development in counting devices that can be traced to the humble beginnings of the abacus. Western civilization seems to have been more interested in the development of counting devices that might use current technology to improve the speed and accuracy of the computation. Thus came the invention of Napier's Bones as a mechanical aid to multiplication. This device was the forerunner of the slide rule which most engineering students used until the proliferation of the electronic calculator. A forerunner of the electronic calculator was an adding and subtracting machine that used gears. This device was invented by Blaise Pascal in 1642 when he was only 18 years old. He invented it to help his father count the taxes that were being collected.

Although computers as you know them have been in existence since the late 1940's, computers existed in the minds of inventors long before they were actually capable of being built. In 1822, Charles Babbage is said to have designed a machine that exhibited the basic components of today's modern computer systems. This machine was called the Analytical Engine. It was programmed by using cards that had holes punched in them. The first programmer of the Analvtical Engine was a woman, Lady Ada Augusta Lovelace, daughter of Lord Byron, the English poet. Unfortunately, both Babbage and Lovelace were ahead of their time. The technology of the day could not support the development of the invention.

The next technological advance came in 1890 when Herman Hollerith, an employee of the U.S. Census Bureau, tackled the problem of speeding up the tabulation of the census reports. Hollerith used a card about the size of a dollar bill, and he defined a method of coding that would allow information to be stored by punching holes in the 80 columns and 12 rows that were on each card. This code is called Hollerith code. In the 1880 census, counting the population of the United States had taken seven and one-half years. Using the Hollerith code and a machine that read and tabulated the information that was on the cards, the 1890 census was done in two and one-half years.

By 1944, the use of electricity fostered the development of the machine that Charles Babbage had envisioned more than a century earlier. The machine was the MARK I, and it was developed by a research team at Harvard University led by Howard Aiken. The MARK I was the first electromechanical computer because it used electromagnetic relays and mechanical counters in it operation. It was followed in 1947 by the ENIAC, the first electronic computer. This computer was developed by a research team under the direction of J. Presper Eckert and John Mauchly at the University of Pennsylvania. As time progressed these machines were improved for commerical use. By 1951, Eckert and Mauchly had produced and sold the first commerical computer called the UNIVAC I. About this time another computer company called IBM was moving from its tabulating business machines into the new world of computers.

The computer world is divided into generations. The first generation is typically between 1942-58, and its machines are characterized by their vacuum tube technology, the great amounts of electricity that they needed, the great amounts of heat that they produced, and the great amounts of area that they occupied. The second generation went from 1959-63. Its machines were characterized by their transistor technology. They required less power, and physical space than the previous generation of machines and they produced less heat. Furthermore, they were much more reliable and did not break down as often. The third generation began in 1963 and ended sometime in the early 1970's. The change in technology that brought about this generation and subsequent generation was the integrated circuit. Integrated circuits were first made with only a few hundred transistors embedded into a small piece of silicon, but today we have integrated circuit with over 100,000 transitors embedded into the same sized piece of silicon. Fourth generation computers came about because of better integrated circuits that were more powerful than those found in third generation computers. Fourth generation computers were made between the early 1970's and about 1980; they are characterized by VLSI circuits (Very Large Scale Integration). We are living in the Fifth generation of computer systems. It will take us a while to look back and decide what will characterize this generation. However, as the experts see it now, it will most likely be the integration of the hardware and software elements of the machine in its original design.

MODULE 2
Introduction to Computer Hardware and Software

Objectives

To demonstrate a knowledge of computer hardware by:
1. Classifying computers by categories of mainframe, minicomputer, and microcomputer.
2. Naming the basic components of a computer -- CPU, Input, Memory, and Output.
3. Explaining the differences between RAM and ROM.
4. Explaining the use of three input and three output devices.
5. Using the computer keyboard.

To demonstrate a knowledge of computer software by:
1. Distinguishing between system software and application software.
2. Identifying the categories of educational software.
3. Explaining the difference between lower level programming languages and higher level programming languages such as BASIC, COBOL, FORTRAN, and Pascal.
4. Identifying three kinds of applications software that would be used in a small business.

Activities

1. Give examples of mainframe, minicomputer, and microcomputers.
2. Given a diagram of a computer, label the parts as CPU, Input, Memory, and Output.
3. Classify each item in a list of memory characteristics as pertaining to either RAM or ROM.
4. Classify a list of devices as either input, output, or input/output device.
5. Compare the keyboards of different brands of computers.
6. Locate the special function keys, numeric keys, and alphabetic keys on the keyboard.
7. Boot the computer system available.
8. Distinguish between administrative, computer assisted instruction, and programming uses of computers in education.
9. Write an output statement in BASIC, COBOL, FORTRAN and Pascal.
10. Explain the educational uses of such application software packages as an electronic spreadsheet, database management system, and a word processor.

Introduction to Computer Hardware and Software

A typical definition of a computer is "an electronic machine that performs rapid, complex computations without much human intervention." Even with precise meanings for the words "rapid," "complex," and the "degree" of human intervention, the notion of what a computer really is remains nebulous. You must look beyond the outside of the machine to understand what a computer is. There are three categories of computers: mainframes, minicomputers, and microcomputers. Although what distinguishes these categories is the cost of the equipment, fundamentally these machines have the same basic components; that is, input, output, memory and a central processing unit (CPU). These basic components are called hardware. Hardware is any physical component of a computer, either mechanical or electronic. On a mainframe computer, there will be card readers, very fast printers, magnetic tape drives, and much more, but on a microcomputer there are keyboards, slow printers, and cassette tape drives. Since microcomputers are the typical computers of education, you will concentrate on the components of this machine. In a microcomputer system, the typical hardware that you can see without physically opening the box consists of a keyboard, a video display screen, a disk storage system, a printer, and other special-purpose devices.

Keyboards are said to have "standard typewriter" features, but the only feature that really is standard is the placement of the alphabetic letters. The special symbol keys such as colon, plus sign, and others are usually in different places on different keyboards. There are also different "special function" keys that appear in different places on the keyboard for different computers. Examples of these are the escape key (ESC), the control key (CTRL), the caps lock keys (CAPS), and the clear screen key (CLEAR). The other point to note is that these keys sometimes take on different meanings when an application program is being run on the computer. In addition to these facts, the keys themselves can be different. Several microcomputers use "chiclet" keyboards. The keys on this microcomputer look like pieces of Chiclet Gum, and if someone is a touch typist, this particular keyboard is very hard to use because the keys are separated by more distance than those on a normal keyboard. However, handicapped students that have the problem of striking too many keys at a time on other keyboards will find the chiclet keyboards easiers to use. Finally, some keyboards have user definable function keys. These usually appear on the keyboard as F1, F2, F3, and so forth. The actual number of special function keys on a keyboard varies. Again, with different applications programs, these special function keys might take on different meanings.

The video display screens are just as varied as the keyboards. You can have white characters on a black background, you can have black letters on a white background, you can have yellow or amber letters on a black background, or you can control the color of the letters and background yourself.

Studies have shown that the green or amber lettering on a black background are easier on the eyes for those people that use the video display screen for long periods of time. The diagonal sizes of the screens also vary, with 4", 9", 12", and 23" diagonals being the most common for monitor types of screens. There are smaller liquid crystal display units on the portable or hand-held microcomputers. In the older versions of microcomputers, the number of characters that could fit onto a line of the screen varied. Some microcomputers displayed 40 characters per line, others displayed 63 characters per line normally, but you could press a couple of keys and get enlarged characters at 32 characters per line. Today, most microcomputer manufacturers have settled on 80 characters per line with 24 lines on a screen. All of these points should be considered very carefully when selecting a video display screen for the visually impaired.

A disk storage system is used to keep information that is not being used by the microcomputer at the present moment. Floppy disk storage devices are the most common on microcomputer systems today; however, many people are starting to need and use the greater storage capacity of the hard disks that may be installed on most microcomputer systems. Presently, floppy diskettes, the medium that is used in a floppy disk storage device, come in sizes of 3 1/4", 5 1/4", and 8". It should be noted that the storage capacity of the floppy diskette is not at all proportional to the physical size of the diskette. When blank floppy diskettes are purchased they must be initialized (formatted) on the computer that is going to use them so that they will be capable of holding information. The initialization process varies from microcomputer to microcomputer but, in general, the floppy diskette is organized into concentric circles called tracks. These tracks are "sliced" like pieces of pie into areas that are called sectors so that information can be retrieved via unique disk storage addresses of track and sector. The information that is stored on the floppy diskette is in magnetic/non-magnetic dots called bits (BInary digiTS). It takes 8 bits to represent any key that you might press on the keyboard. Technically, 8 bits are called a byte but in common usage it is often called a character. For ease of calculation, bytes are grouped together in a measurement called K. A K or kilo is approximately 1000 and exactly 1024. This latter number has to do with a power of two which represents the bit because it can be either on or off. Thus, if you say that the capacity of your floppy diskette is 144K, you are implying that it can store approximately 144,000 characters, but you know that it actually stores a little more, 147,456 characters. Again, the storage capacity of an actual diskette depends on the microcomputer on which it is to be initialized; furthermore, today some microcomputers use both sides of the diskettes (double sided as opposed to single sided), and they compact the dots closer together (double density as opposed to single density). The storage principles behind the hard disk are those of the floppy disk storage device with the exception that the hard disk cannot be removed by the user of the microcomputer. As a result, hard disks are more precisely tooled for higher density and faster disk rotational speeds. This all makes for a larger storage capacity, in fact, millions of bytes or mega-bytes which are symbolized by MB. Common capacities for hard disk units on microcomputers are 5MB, 10MB, and 20MB. Although hard disks can make educational computer systems more desirable for special educational uses, they are more expensive and it takes longer to copy the data stored on the hard disk (because there is so much of it). This activity of making copies of your programs and information that you have stored on a disk storage device is called making a backup. It is a good idea to back up both floppy systems and hard disk systems periodically.

Printers can be relatively expensive items. They vary in the quality of their printing, their speed, and their capacity for a line of writing. You could spend hours upon hours discussing the different kinds of printers that can be found in today's marketplace. Educational printers tend to be dot matrix printers. These printers are relatively fast and reliable with heavy usage. The characters that are formed by these printers are literally formed by dots. In Figure 1 that is attached to this module, you can see an enlarged dot matrix printing at the top followed by a normal dot matrix printing of the same information. The visually impaired student might find the former printing easier to read than the latter one. The third type of printing is called correspondence quality, and it is compressed dot matrix. That is, the dots are put so close together that they do not seem to be dots at all. The final example is that of letter quality. Letter quality printers tend to be slower than the dot matrix printers and the size of their characters are fixed. Larger type fonts are available. One final printer that should be mentioned is the graphics printer. Graphics printers are usually dot matrix printers that have a greater capacity for controlling their printing dots; thus, drawings are easy to produce. Attached to this module is Figure 2 which was printed on a graphics printer.

The keyboard is an example of an input device, the video display screen and the printer are examples of output devices, and the disk storage devices are examples of input and output devices. In addition to these common pieces of hardware, there are a great many different kinds of special purpose devices that you can have on a microcomputer system. These pieces of hardware are also characterized as input, output, and both input and output devices. Examples of special purpose input devices are card reader, light pen, mark sense reader, joystick, mouse, graphics tablet, voice recognition, and so forth. Examples of other special purpose devices that are used for ouput are plotters and voice synthesizers. Input and output special purpose devices are magnetic cassette recorders and modems. Special needs require special equipment, so a typical microcomputer configuration in one environment may not be a typical microcomputer configuration in another environment. The beauty of microcomputers is that they can be tailored to fit special needs.

Even after noting all of these devices, you are still not finished with the hardware because for once, what you see is not all that you really get. If you were to open the microcomputer and look inside (NOT RECOMMENDED! This might void any warranty.) then you might be surprised to see that there is a lot of air in the box. The largest amount of hardware that you would see inside the box would be the power supply unit. This is the "heart" of the microcomputer because without electrical power, a microcomputer is just an expensive paper weight. The "brain" of the microcomputer is its central processing unit (CPU). The CPU is a chip of silicon less that a quarter of an inch square. It contains the equivalent of tens of thousands of discrete electrical components such as transistors, capacitors, resistors, and diodes. Also inside the microcomputer, are rows and rows of memory chips. There are two types: Read Only Memory (ROM) and Random Access Memory (RAM). ROM is permanent memory, and it is not lost when the microcomputer loses electrical power. The information in ROM is fixed by the microcomputer manufacturer. Usually ROM tells the microcomputer what to do as soon as it is turned on; however, on smaller microcomputers ROM also contains the programming language of BASIC. RAM is temporary memory. It can be erased and altered at will. It vanishes completely if power is cut off even for a fraction of a second. The most important thing to remember is that the amount of RAM determines how large an application program you can have on your microcomputer. RAM is also measured by K. Early microcomputers (circa 1976) had 4K RAM but today the minimum amount of RAM sold is 64K with upper limits going beyond 512K.

Although much time and effort has been spent on the hardware aspect of microcomputers in this module, hardware without software is just as useless as hardware without electrical power. Software is the instructions necessary to have a microcomputer perform a useful task. There are two general types of software: system software and application software. System software is of less direct concern to you, except insofar as it determines what applications software can be run. System software is often copied into the microcomputer's RAM from a diskette when the microcomputer is turned on; however, on smaller microcomputers system software is found in ROM. Examples of system software are DOS (Disk Operating System) and CP/M (Control Program for Microcomputer). Each microcomputer manufacturer has its own system software; for example, some Tandy computers use LDOS while others use MS-DOS. IBM has PC DOS and MS-DOS for its microcomputer. Some manufacturers would like to see system software standardized, but the chances of that happening at the moment are slim.

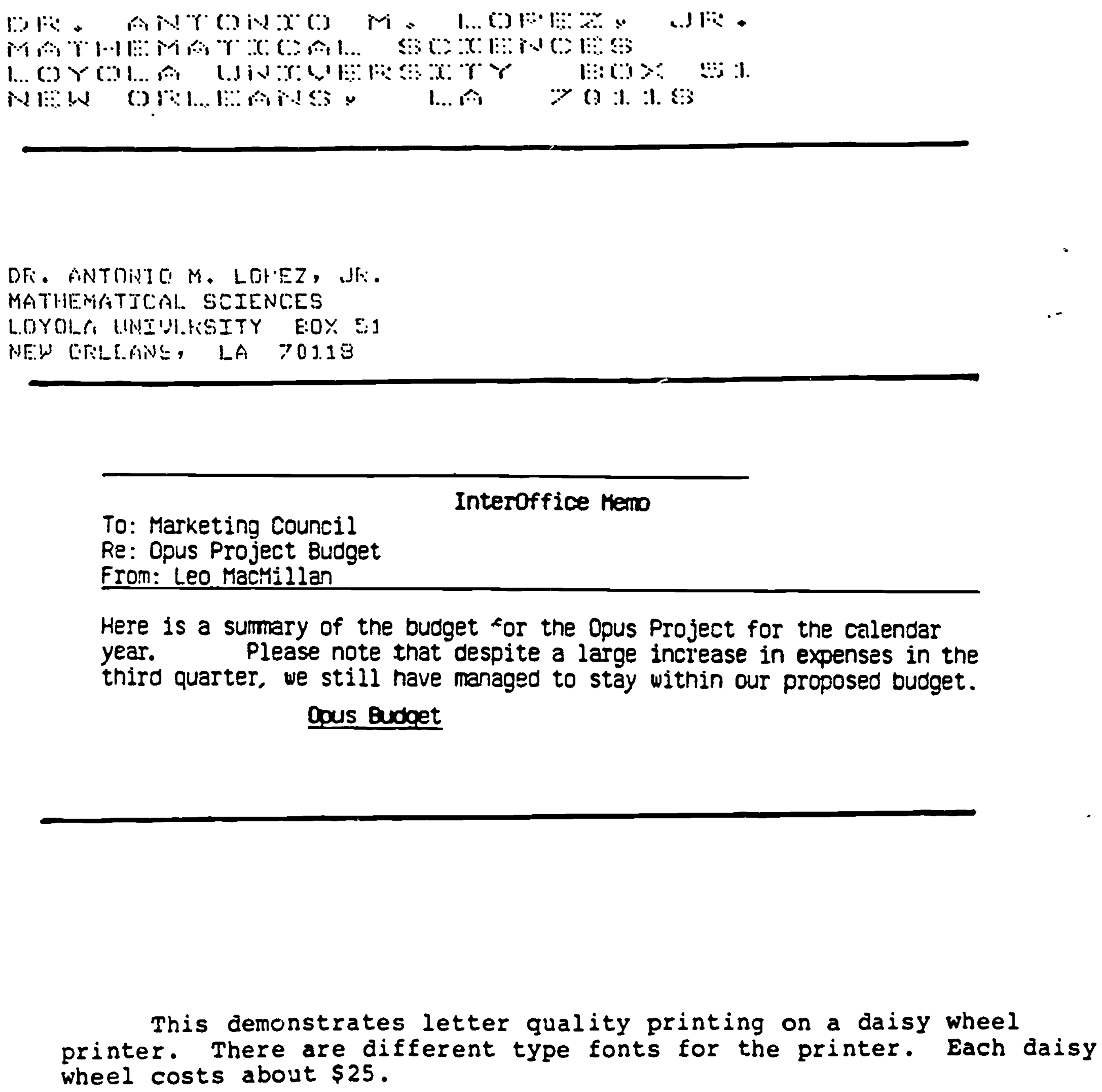
Since the purchase of a particular kind of microcomputer hardware implies the purchase of the system software necessary to use that hardware, you will naturally be more concerned with the application software than system software. Application software is a series of instructions written in a programming language which performs a useful task such as allowing you to program the computer in a high level programming language such as BASIC or allowing you to use the computer as a word processor to write papers. Presently, the application software that is being used on microcomputers in an educational environment fall into three general categories: Administrative Applications, Computer Assisted Instruction, and Programming Languages.

Administrative software is software that is being adapted for educational uses mainly from commerically available business software. Examples of this kind of software are electronic spreadsheets, database management packages, and word processors. With electronic spreadsheet, software classroom teachers can better manage their classes by maintaining gradebooks with current class averages as well as current individual averages. With database management software, principals can store and retrieve information concerning permanent teachers, subsitute teachers, and job applicants. Furthermore, this software can help them produce more accurate reports for local, state, and federal officials. However, without doubt the number one administrative application software package has to be word processing. Using a word processor, you can produce "individualized" school correspondence that is letter perfect for each student's parents. Articles can be prepared for publication and so can books. The text for a document appears on the video display screen as it would appear if it had been typed with a typewriter; however, before printing the document misspellings can be corrected, paragraphs moved, word structures altered, and the entire document repaginated. Once you have used a word processing software with your microcomputer, you will throw away your typewriter and with it your white-out.

There are many kinds of computer-assisted instruction (CAI) software but most educators would agree to classify the different kinds as Drill and Practice, Tutorial, Testing, Simulation, and Dialog. Drill and Practice software presents exercises to the student to reinforce learning gained from another source--the teacher. The approach is strictly a supplemental one. The teacher introduces and explains new concepts and the microcomputer is used to exercise the students understanding of those concepts by providing exercises. This kind of software should provide the student with immediate feedback after the student has answered a question. It should also keep track of how the student in doing on that particular lesson so that a score can be presented at the end of the drill. Tutorial software is similar to the drill and practice software with the exception that the tutorial software goes beyond just giving the student the correct answer when an incorrect response has been made. Tutorial software attempts to explain to the student when and how the error was made. Testing software allows teachers to compose tests for their students. These tests may then be printed out on a ditto master, or students can take their "personalized" version on the microcomputer. Basically, the teacher generates a pool of questions from which the computer selects a certain number at random. If the test is printed out, then the answer sheet is also printed. If the test is taken on the computer then the computer grades the student's responses. There are a number of good authoring programs on the market today, but they usually can produce only true/false, multiple choice, or matching questions. At present there are no testing programs that will read and evaluate an English paper. Simulation programs allow microcomputers to imitate real situations in a physical or social system. In this way, students may observe how changing different variables will affect the overall operation of the system. An administrative application software package that is also a good simulation package is the electronic spreadsheet. The dialog software allows students to interact with a microcomputer and discover for themselves the lesson that is being taught. The student is not drilled, tutored, or tested. Instead, the computer questions the student about general items of interest and based upon the student's response, the computer produces another question that will lead the student closer to the subject that is to be taught. Dialog software is very difficult to write, and as a result, there is not too much of it available for microcomputers. However, the PLATO system which was originally designed to run on mainframes has already been moved to minicomputers, and there are projects under way to move certain topics to microcomputers.

The last kind of educational software to be considered is programming languages. The lowest level of programming a computer is machine language programming. In this language it is important to understand bits, bytes, and hexadecimal coding. The advantage of machine language programming is that its execution by the computer is very fast since there is no translation needed; that is, you are programming in the langauage that is directly understood by the machine that you are using. You should note that the machine language for a microcomputer that has a 6502 microprocessor as its CPU is different from the machine language for a microcomputer that has a 68000 microprocessor as its CPU. One step higher in the hierarchy of programming languages is assembly language programming. Assembly language varies from machine to machine just like machine language but instead of programming with bits, bytes, and hexadecimal codes, assembly language allows you to use mnemonics; letter codes and labels that represent hexadecimal codes and memory addresses. For example, in machine language on a 6809 microprocessor a GOTO a user-defined function at memory location 0300 would be 7E 0300 and in assembly language this would be JMP USR1. The advantage of assembly language coding is that it is easier for the programmer to remember the mnemonic JMP for "jump to" instead of the hexadecimal code 7E which does the same thing. Further removed from machine language are the high level programming languages. Some examples of these are BASIC, FORTRAN, COBOL, and Pascal. Each of these languages has its own colorful history, and the first three are so called because they are acronyms; BASIC stands for Beginners All Purpose Symbol Instruction Code; FORTRAN stands for FORmula TRANslation; and COBOL stands for COmmon Business Oriented Language. Pascal is a language that was named in honor of Blaise Pascal. Each of these languages was developed with a special purpose in mind so it would not be appropriate to say that there is one best high-level programming language. Furthermore, there are many more high level programming languages than have been listed here. In fact, at last count there were well over 70 high level langauges being used throughout the computer industry today. As a particular example, you should note the work of Seymour Papert with his language of LOGO. LOGO was invented at MIT for the purpose of teaching very small children who could not even read or write how to use the computer to learn more about their world. LOGO uses a symbol on the video display screen called a "turtle" and allows the child to command the activities of the turtle. By entering the word FORWARD 10 or the abbreviation FD 10 you cause the turtle to move 10 units forward. To turn the turtle to the right 45 degrees, the child enters RIGHT 45 or RT 45. There have been some very successful stories told about the use of this programming language with handicapped students.

As a final point in this module, you are asked to look at Figure 3 which is attached. This figure gives you the different views of application software. The programmer see the programming instructions that are necessary for the computer to preform a useful task for the user. The programmer is also aware of the support that is necessary from the system software and the programming language software that is being used. (Your example is written in BASIC.) On the other hand, the user of the application software sees only the questions that are being asked and it is hoped the user will respond correctly to those questions. After the user has responded, the computer saves the user a lot of time by calculating, manipulating, and organizing the necessary facts for a subsequent report. All of this and all that was covered previously must be understood in order for a user to properly evaluate an application software package. The evaluation procedure will be covered latter in Module 4.



DR. ANTONIO M. LOPEZ, JR.
MATHEMATICAL SCIENCES
LOYOLA UNIVERSITY BOX 51
NEW ORLEANS, LA 70118

DR. ANTONIO M. LOPEZ, JR.
MATHEMATICAL SCIENCES
LOYOLA UNIVERSITY BOX 51
NEW ORLEANS, LA 70118

InterOffice Memo

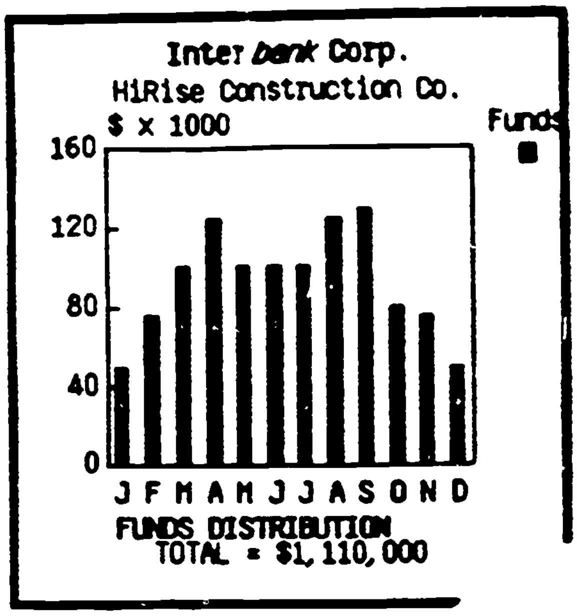
To: Marketing Council
Re: Opus Project Budget
From: Leo MacMillan

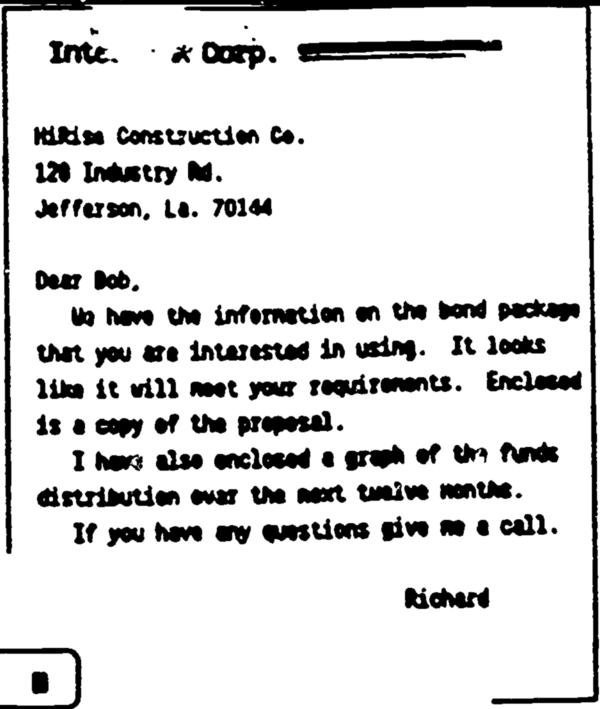
Here is a summary of the budget for the Opus Project for the calendar year. Please note that despite a large increase in expenses in the third quarter, we still have managed to stay within our proposed budget.

Opus Budget

This demonstrates letter quality printing on a daisy wheel printer. There are different type fonts for the printer. Each daisy wheel costs about $25.

Figure 1




Inte. ·k Corp.

HiRise Construction Co.
120 Industry Rd.
Jefferson, La. 70144

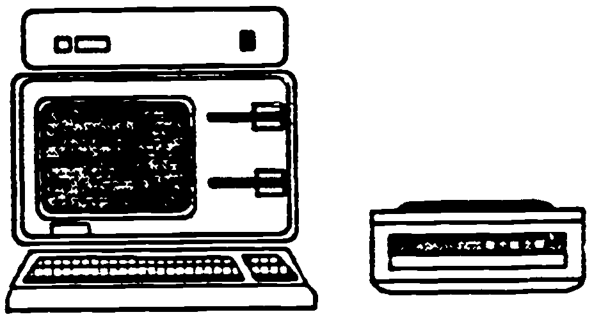
Dear Bob,

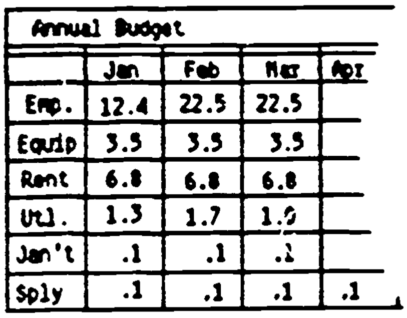
We have the information on the bond package that you are interested in using. It looks like it will meet your requirements. Enclosed is a copy of the proposal.

I have also enclosed a graph of the funds distribution over the next twelve months.

If you have any questions give me a call.

Richard

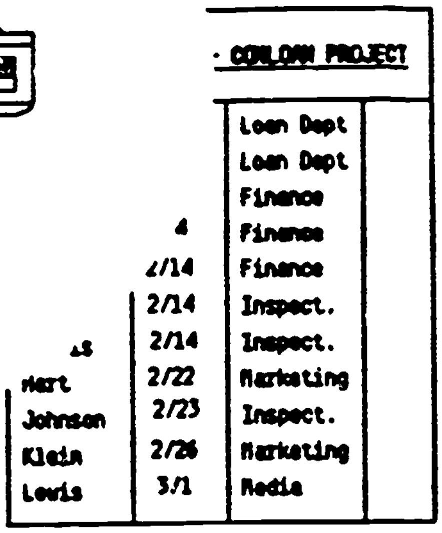



Annual Budget

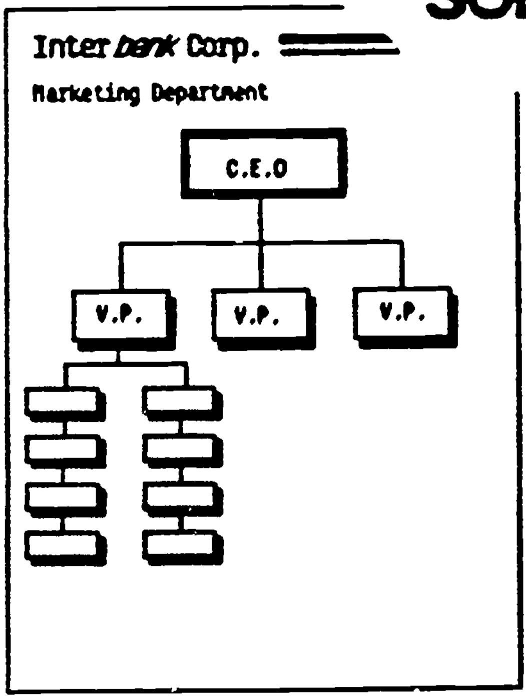
	Jan	Feb	Mar	Apr
Emp.	12.4	22.5	22.5	
Equip	3.5	3.5	3.5	
Rent	6.8	6.8	6.8	
Utl.	1.3	1.7	1.9	
Jan't	.1	.1	.1	
Sply	.1	.1	.1	.1

Lisa

SOLUTIONS


· CONLOAN PROJECT

		Loan Dept
		Loan Dept
		Finance
	4	Finance
	∠/14	Finance
	2/14	Inspect.
..s	2/14	Inspect.
Mart	2/22	Marketing
Johnson	2/23	Inspect.
Klein	2/26	Marketing
Lewis	3/1	Media



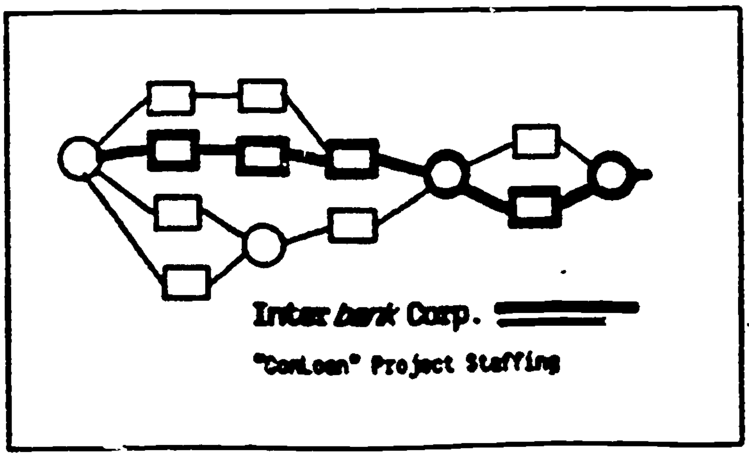


Figure 2

BEST COPY AVAILABLE

PROGRAMMER VIEW:

```
1       REM       DEPRECIATION PROGRAM
10      CLS:INPUT"WHAT IS THE COST OF YOUR ASSET";C
20      INPUT"WHAT IS THE USEFUL LIFE IN YEARS";N
30      INPUT"WHAT IS THE EXPECTED SALVAGE VALUE";S
40      A=(C-S)/N
50      I=N
55      CLS:PRINTTAB(20)"DEPRECIATION SCHEDULE":PRINT"YEAR";TAB(10)"STRAIGHT LI
";TAB(30)"SUM OF YEARS DIGITS"
60      FOR Y=1 TO N
70      D=(2*I*(C-S))/(N*(N+1))
80      PRINTTAB(2)Y;TAB(13)A;TAB(35)D
90      I=I-1
95      NEXT Y
100     END
```

PROGRAM INSTRUCTIONS.

USER VIEW:

QUESTIONS ARE ASKED BY THE PROGRAM.

QUESTIONS ANSWERED BY THE USER.

```
WHAT IS THE COST OF YOUR ASSET?  10000
WHAT IS THE USEFUL LIFE IN YEARS? 10
WHAT IS THE EXPECTED SALVAGE VALUE?  500


                    DEPRECIATION SCHEDULE
YEAR       STRAIGHT LINE        SUM OF YEARS DIGITS
  1            950                  1727.27
  2            950                  1554.55
  3            950                  1381.82
  4            950                  1209.09
  5            950                  1036.36
  6            950                  863.636
  7            950                  690.909
  8            950                  518.182
  9            950                  345.455
  10           950                  172.727
```

REPORT PRODUCED BY THE PROGRAM FOR THE USER

THE MANY VIEWS OF SOFTWARE

FIGURE 3

MODULE 3
Introduction to Computer Programming with BASIC

Objectives

To demonstrate understanding of BASIC programming by:
1. Using the computer in the Command, Edit, and Run modes,
2. Entering instructions, erasing, listing, and running programs,
3. Using arithmetic operators and understanding their hierarchy of operation,
4. Understanding the three basic programming constructs of sequence, selection, and looping,
5. Analyzing a given problem and writing the steps in this solution so that a computer can execute the solution,
6. Writing simple programs using LET, INPUT, READ, DATA, PRINT, GOTO, IF...THEN..., FOR...NEXT, and END.

Activities

1. Display a string of characters by using the PRINT command.
2. Identify the line number, BASIC keywords, and variables in a program.
3. Debug and modify an existing program by adding or deleting lines.
4. Assign values to variables using the LET and INPUT statements.
5. Replace two or more LET statements with INPUT statements.
6. Design an interactive program using INPUT and PRINT statements.
7. Write a simple looping program with a FOR...NEXT.
8. Translate a set of problem solution steps to the corresponding BASIC program statements.
9. Write a selection program using an IF...THEN...
10. Evaluate an arithmetic expression which includes a combination of two or more arithmetic operations.

In general, there are three operating modes for a computer: the Command mode, the Edit mode, and the Run mode. In the Command mode, the computer responds to instructions as soon as they are typed in to the computer and the ENTER or RETURN key is pressed. In the Edit mode, the computer does not respond to instructions that are entered but takes those instructions and places them into RAM for further reference and eventual execution in the Run mode. Instructions in the Command mode do not have line numbers, and instructions entered in the Edit mode have line numbers. The Run mode takes those instructions in RAM and executes them starting with the lowest line number that was entered regardless of when it was entered.

Example 1: Enter each of the following into the computer:

```
PRINT 5 + 10

PRINT "DISPLAY THIS NOW!"
```

The arithmetic operators that can be used in computers are + for addition, - for subtration, * for multipilcation, and / for division. The order of operation is multiplication and division first, moving from left to right and then addition and subtraction also moving from left to right. Parentheses may be used to alter this order because whatever is enclosed in parentheses is calculated first. Besides the arithmetic operators, computers have a number of "built-in" functions much like our present day calculators. These include the trigonometric, the logarithmic, and the square root functions, as well as many more.

Example 2: Enter each of the following into the computer:

```
PRINT SIN(0)

PRINT ABS(5 - 30)

PRINT RND(.5)
```

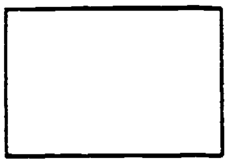
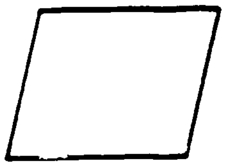
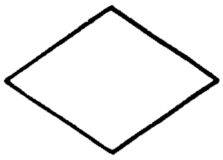
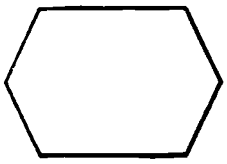
The Command mode has three instructions that are used in conjunction with activities that are being done in the other two modes. The command NEW is used before typing in a program into RAM to erase any old program that might be in RAM. The command LIST is used to display the program that is in RAM at the time. The command RUN causes the computer to do whatever the program that is in RAM at the time tells it to do.

The Edit mode can be simply illustrated by typing a line number and one of the statements in the previous examples. Once that has been done nothing will happen that you can see. The instruction that you typed into the computer has been placed into RAM for later execution when you type the word RUN (which gets the computer into the Run mode) and press the ENTER key.

Example 3: Enter and RUN the following:

```
10 PRINT "A RANDOM NUMBER BETWEEN 0 AND 1 IS"
20 PRINT RND(.5)
```

As long as we have single step "thoughts" that we want accomplished by the computer, the programming effort is not very complicated. However, the depth of human thought is far from a single step. As a result we need to connect single step thoughts together to form more complicated procedures that a computer can follow. We can do this graphically with the understanding that with each graphic symbol there is associated at least one computer instruction. Throughout this module we will use a graphic method to demonstrate the "flow" of a program. This method is called flowcharting, and the symbols that we will use are as follows:

Flowchart symbol	BASIC statement(s)
	LET
	PRINT INPUT READ
	IF ... THEN ...
	FOR ... NEXT ...

These blocks are connected by directional arrows called flow arrows.

In programming there are three basic constructs -- sequential, selection, and looping. In a sequential program, when a RUN command is issued the computer begins executing program instructions found in RAM from the lowest numbered line in ascending order. In BASIC, a variety of instructions can be used to implement a sequential construct.

Example 4: Enter and RUN the following:

```
10 LET A = 2
20 LET B = 6
30 LET C = A + B
40 PRINT "THE SUM OF "; A;" AND "; B;" IS "; C
50 END
```

Example 5: Enter and RUN the following:

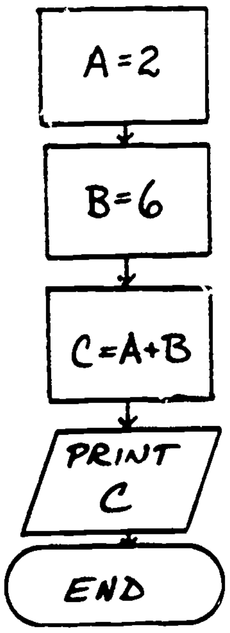
```
10 INPUT "ENTER A NUMBER ";A
20 INPUT "ENTER ANOTHER NUMBER ";B
30 LET C = A + B
40 PRINT "THE SUM OF "; A;" AND "; B;" IS "; C
50 END
```

Example 6: Enter and RUN the following:

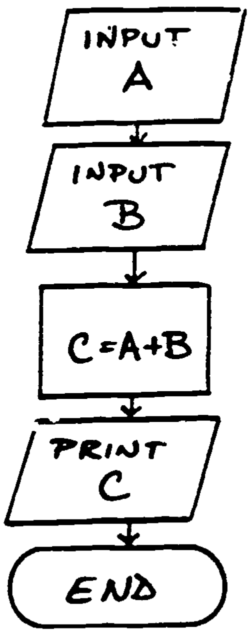
```
10 READ A,B
20 DATA 2, 6
30 LET C = A + B
40 PRINT 'THE SUM OF "; A;" AND "; B;" IS "; C
50 END
```

The flowcharts to each of these programs look very familiar.

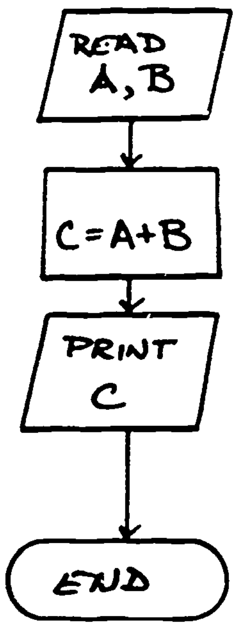
Example 4



Example 5



Example 6

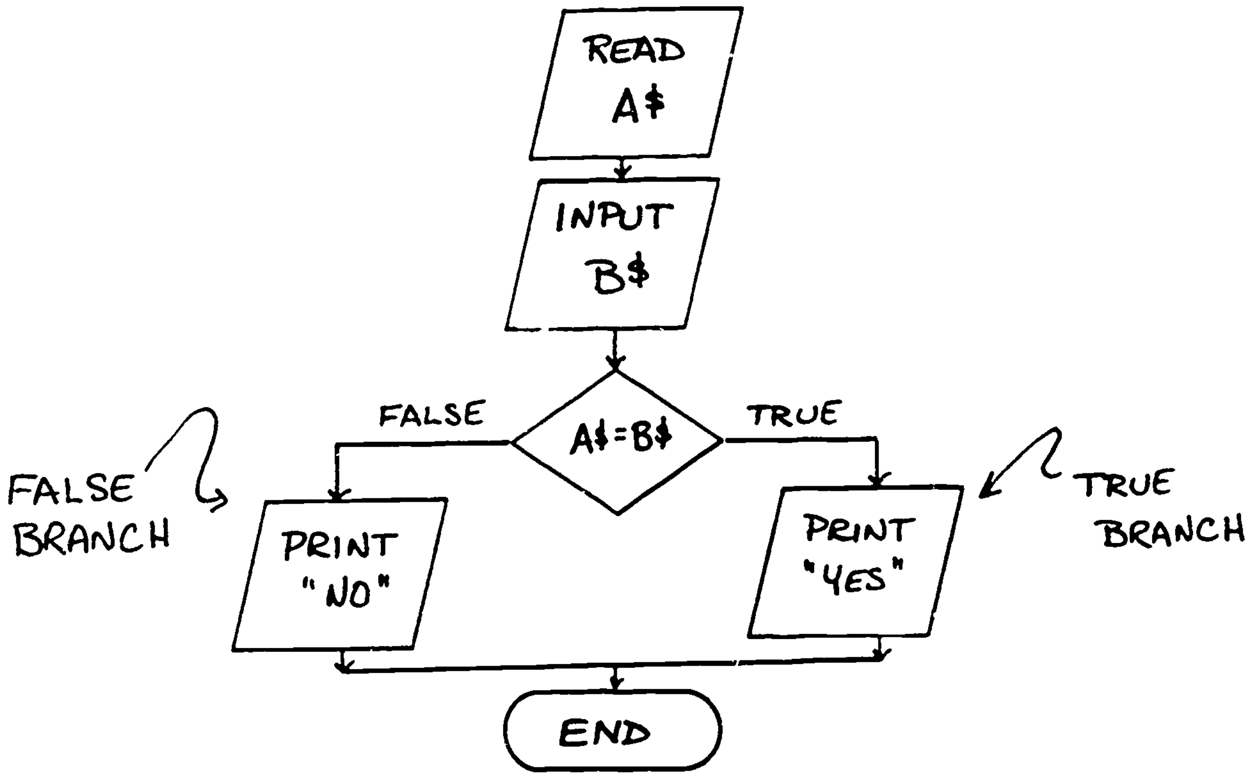


The selection structure represents a branch point at which the course of execution depends upon some specific condition. The selection structure always has two exits--a TRUE branch and a FALSE branch. In BASIC, the selection construct is implemented by the IF ... THEN ... instruction.

Example 7: Enter and RUN the following:

```
10 READ A$
20 DATA ELEPHANT
30 PRINT "I AM THINKING OF AN ANIMAL."
40 INPUT "WHAT ANIMAL IS IT";B$
50 IF A$ = B$ THEN GOTO 80
60 PRINT B$;"??? NO, THAT'S NOT IT!"
70 END
80 PRINT "YES, THAT'S IT. LUCKY GUESS!"
90 GOTO 70
```

The flowchart for this program is



The idea of repeating a certain set of instructions is called a loop. It is an idea that is central to computer programming and is probably the single most important concept invovled in understanding programs. Each loop structure has a counter that counts how many times what is in the loop is to be performed. What is in the loop structure is called the body of the loop. Any statement or structure may appear in the body of a loop. In BASIC, the looping construct is implemented by the FOR ... NEXT set of instructions.

Example 8: Enter and RUN the following:

```
10 FOR N = 1 TO 10
20 PRINT N
30 NEXT N
40 END
```

Example 9: Enter and RUN the following:

```
10 INPUT "ENTER A POSITIVE NUMBER LESS THAN 200";M
20 PRINT "THE SUM OF THE NUMBERS FROM 1 TO ";M
30 LET S = 0
40 FOR N = 1 TO M
50 LET S = S + N
60 NEXT N
70 PRINT "IS ";S
80 END
```

In order to do something substantial with the instructions that we have learned, we must be able to write programs that will perform some meaningful task; that is, we must be able to solve a particular problem.

PROBLEM-SOLVING PROCEDURE:

Step 1. Get a clear understanding of the problem.
Although this is probably the most obvious step in problem solving, it is the one that gives the greatest amount of difficulty. All computer problems are word problems, so you must clearly determine the givens and what is to be found. In computer terminology, this is called INPUT and OUTPUT respectively.

Step 2. Select a method of solution.
There may be several ways of getting from INPUT to OUTPUT, in which case you must now select the method that you would like to employ. On the other hand, there may be only one way of getting from INPUT to OUTPUT, so you may not have any alternative. Finally, there may be no method known to you on how to get from INPUT to OUTPUT, so you must realize that the computer cannot help you since it can only do what you tell it to do. There are a variety of names given to this particular problem-solving step; some are PROCESSING, PSEUDOCODE, STRUCTURED ENGLISH, ALGORITHM, and so forth.

Step 3. Draw a picture of your solution.
Actually, you want to refine the structure of your solution presented in Step 2, so you draw a "road map" to follow in a step-by-step journey from INPUT to OUTPUT. Successful completion of this step guarantees that you have solved the problem in such a fashion that the solution can be implemented on a computer. There are several ways of drawing pictures of your solution. One such way is a FLOWCHART. A FLOWCHART is helpful when a problem

is relatively small and you just want to document your progression from one instruction to the next. Large "real world" problems make FLOWCHARTS impractical except maybe to explain some intricate point of the solution. In such cases, you might use a HIERARCHY CHART, a DESIGN DIAGRAM, or a DATA FLOW DIAGRAM. Since the problems that you will be solving are relatively small, you will be trained on FLOWCHARTS.

Step 4. Convert the picture into a computer language.
This step is better know as PROGRAMMING or CODING. A program is a picture that has been translated into a particular programming language. It may surprise you at this point, but the hard part of the programming process is the development of the picture. You will be converting your FLOWCHART into the programming language of BASIC. Once you have learned the "vocabulary" and the "grammatical rules" of BASIC (an easy task compared with learning a human foreign language), the translation of the picture is quite straightforward. It is very important that you separate the two stages of this procedure, and not try to translate into BASIC until you know exactly what it is that has to be done, that is, Steps 1, 2, and 3. A common mistake for beginners, which makes writing larger "real world" programs more difficult than it has to be, is to try to skip Steps 1, 2, and 3 and go to Step 4 immediately.

Step 5. Run the program on the computer and check the answer.
Two kinds of errors can occur when running a computer program: syntax errors and logic errors. Syntax errors are the easiest to handle since the computer itself will indicate that something is wrong with a particular line of the program. Syntax errors occur because you have violated the vocabulary or the grammatical rules of the particular language that you are using. Logic errors are usually more difficult to find. You know that you have a logic error when the answer that the computer gives you is not the answer that you expected. This is somewhat like asking the computer to sum 2 and 3 and having the computer say that it is 6. Something went wrong in the program. In running any computer program, you must make up some TEST DATA that you will provide as INPUT to the program. This TEST DATA should be chosen because you know the OUTPUT for it so if the computer does not give you the OUTPUT that you expected, then you have a logic error in your program. If you have a logic error in your program, you may have to reconsider your progression from Step 1 to Step 4 in this problem-solving procedure. Either you made a mistake in one of these steps or your TEST DATA is invalid. The process of eliminating syntax or logic errors from your program is called DEBUGGING. DEBUGGING is really an art, and the more you do it the better you become at it.

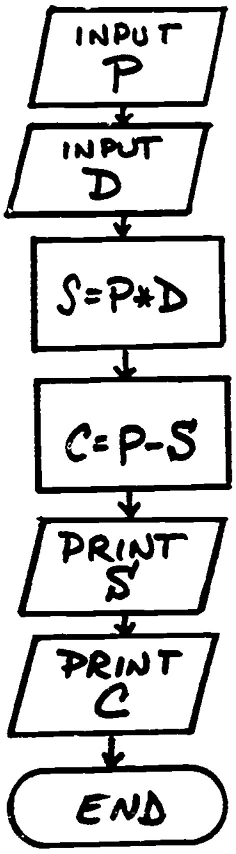
Example 10: Write a BASIC program that will accept from the user a price of an item and a discount rate, calculate and display the amount of discount and the amount paid for the item.
Solution:

```
Step 1.    INPUT:  Price of the item (P)
                   Discount rate (D)

           OUTPUT: Amount of discount (S)
                   Amount paid (C)
```

Step 2.
Upon request the user will enter the price of the item (P).
Upon request the user will enter the rate of discount (D) as a decimal value; in other words, a 50% discount will be entered as .5.
The amount of discount (S) is calculated by taking the product of the price of the item (P) and the rate of discount (D). This is, S=P*D.
The amount paid for the item (C) is calculated by taking the amount of discount (S) and subtracting it from the price of the item (P). This is, C=P-S.
Display the amount of discount (S).
Display the amount paid (C).
End the program.

Step 3.



Step 4.

```
10 INPUT "ENTER THE PRICE OF THE ITEM ";P
20 INPUT "ENTER THE RATE OF DISCOUNT AS A DECIMAL NUMBER ";D
30 LET S = P * D
```

```
40 LET C = P - S
50 PRINT "THE AMOUNT OF DISCOUNT IS ";S
60 PRINT "THE AMOUNT PAID IS ";C
70 END
```

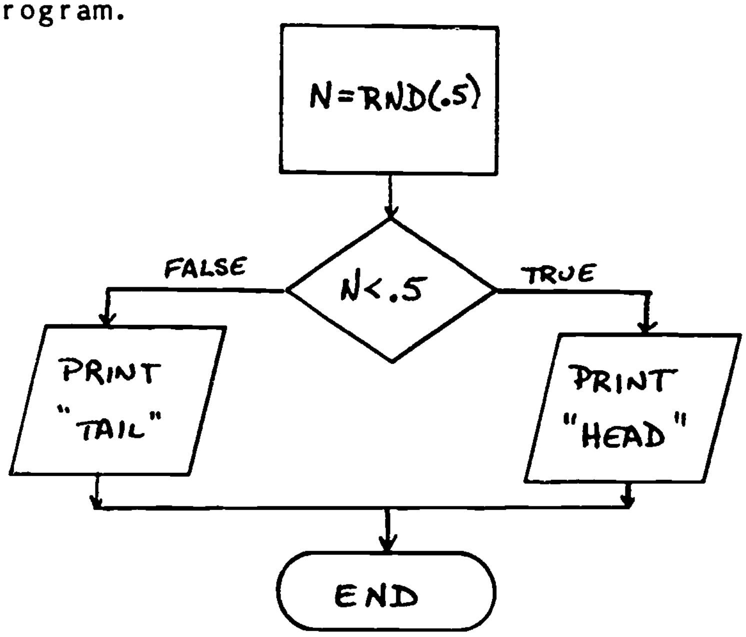
Step 5.
As INPUT use 100 for the price and .25 for the rate of discount. Consequently, you would expect the amount of discount to be 25 and the amount paid to be 75.

Example 11: Write a BASIC program to simulate a flip of a coin.
Solution:

Step 1. INPUT: None from the user.
OUTPUT: A statement that it was a head or a tail.

Step 2.
Invoke the built-in function to get a random number between 0 and 1.
If this number is less than .5, call it a head; otherwise, call it a tail.
End the program.

Step 3.



Step 4.

```
10 LET N = RND(.5)
20 IF N < .5 THEN GOTO 50
30 PRINT "IT WAS A TAIL."
40 END
50 PRINT "IT WAS A HEAD."
60 GOTO 40
```

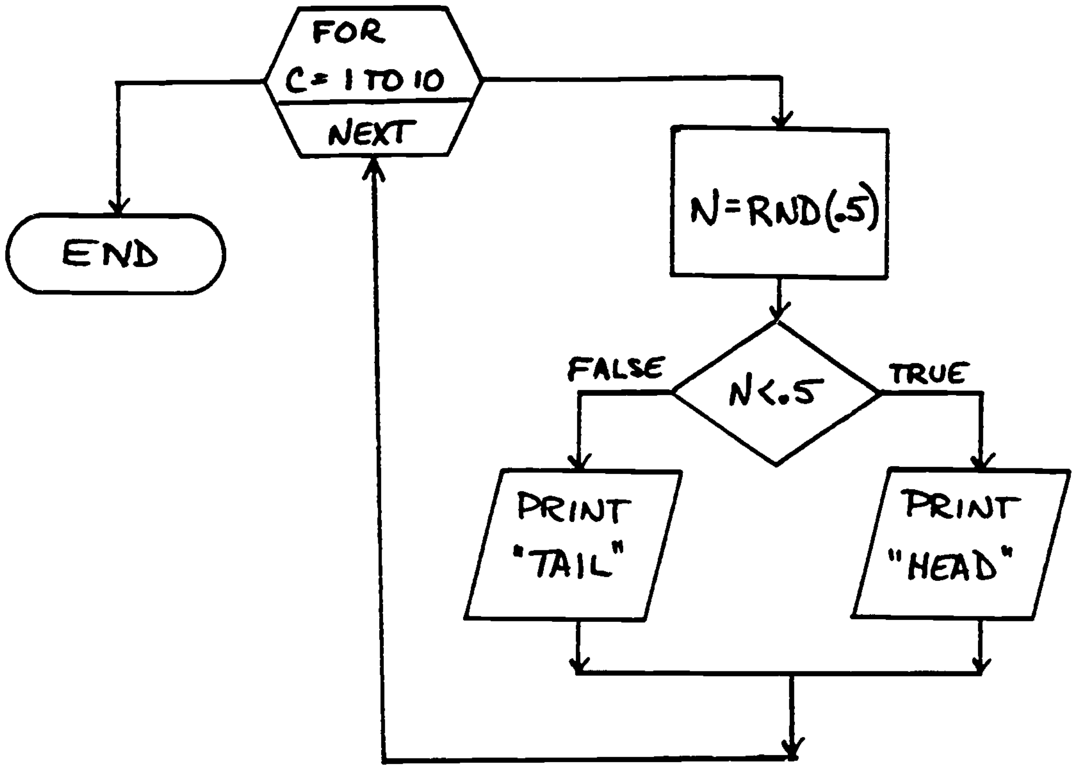
Step 5. Run this program several times so that you can see that both heads and tails are being produced.

Example 12: Modify the BASIC program in the previous example so that it simulates the flip of the coin 10 times.
Solution:

Step 1.
INPUT: None from the user.
OUTPUT: Ten statements, one for each flip, indicating the outcome.

Step 2.
Introduce into the previous example a loop that counts the 10 times that this simulation of a flip is going to occur.

Step 3.



Step 4.

```
10 FOR C = 1 TO 10
20 LET N = RND(.5)
30 IF N < .5 THEN GOTO 60
40 PRINT "IT WAS A TAIL."
50 GOTO 70
60 PRINT "IT WAS A HEAD."
70 NEXT C
80 END
```

There are, of course, more difficult programs, and these basic constructs are used to make more complicated structures. This training module has been designed to introduce you to the world of BASIC programming and not to make you BASIC programmers.

MODULE 4
Evaluating Application Software

Objectives

To demonstrate an ability to assess the need for a particular application software package.

To demonstrate an understanding of application software requirements.

To demostrate an awareness of ethical and legal problems caused by copying application software packages.

To demonstrate an ability to evaluate the appropriateness of a particular software package for a particular group of students in a particular environment.

Activities

1. Given a popular application software package, explain its purpose and possible uses.
2. List sources from which application software can be obtained.
3. After examining an application software package, rate the following:
    (a) readability of screen format
    (b) number of keystrokes for control functions
    (c) readability of report formats
4. For a particular application software package, state the requirements for:
    (a) additional interface cards
    (b) video screen size
    (c) printer type and size
    (d) system software
    (e) main memory
    (f) disk storage
    (g) other peripheral devices
5. List at least two sources of public domain software.
6. List reasons for copyright laws and explain how these laws affect application software.
7. Examine an educational software package and determine its appropriateness for a particular grade and subject.

Evaluating Application Software

There is a big difference between buying software for today's mainframes and today's microcomputers. You can liken buying software for a mainframe to buying a house, while buying software for a microcomputer is more like buying a car. The clear difference is cost, but there is also the important difference of access and distribution. Furthermore, mainframe purchases have life expectancies of 10 years, while microcomputer software might last a year or two at best before the next generation is available. For example, in 1982 VISICALC was the leading electronic spreadsheet program, but by 1983 LOTUS 1-2-3 had taken over the lead. These differences notwithstanding, evaluative procedures for application ftware have been around for a long time and so have the problems. Many years ago a cartoon appeared in a magazine concerning the design, development, and implementation of application software for the large computer mainframes of the period. The cartoon is worth resurrecting because it provides a charming and humorous insight as to why it is so difficult to get the right software. This cartoon classic is called "The Swing," and it appears as Figure 1 in this module.

Believe it or not problems with software are pretty much the same now as they were 20 years ago except now more people are being exposed to the software problems and, of course, are forming their own opinions as to the usefulness of the software. Software evaluation involves three fundamental considerations: Hardware, Software, and People.

In the evaluative process, the hardware consideration is rather straightforward. You either have the microcomputer configuration that will run the software package or you do not. This is probably the single most frustrating aspect of software evaluation because schools are usually being limited to the type of hardware that can be purchased. Somehow it always seems as if the software package that you saw and liked at an inservice workshop that you attended does not work on the microcomputer that you have in your school. This situation is very common. If you were to list the top 10 educational software packages by virtue of sales alone, you would see that four different microcomputers systems would be required to assure you of the hardware capability of running them all. Some software manufacturers have realized this problem and are providing particular versions of educational software packages for each of the four leading microcomputer manufacturers. However, supporting multiple versions of educational software for a variety of hardware is a costly venture for the smaller software manufacturers, and the profits from the educational software sometimes do not justify the effort. Consequently, a very nice reading package that runs on the IBM PC might not have a version that runs on the APPLE IIe. It is also important to understand that even though the floppy diskettes for two different brands of microcomputers may be the same physical size, the information on a diskette for one microcomputer is formatted differently from that of the other. For example, a floppy diskette that operates on the TRS-80 Model 4 will not operate on the Commodore 64 even though they are both 5 1/4" floppy diskettes.

Software considerations for an application package are the system software under which the application package will operate and the programming language in which the application is written. If a disk system is required to use the application, does the application require the support of the hardware manufacturer's system software or does it use an operating system that is available for that hardware but from a different manufacturer? For example, the TRS-80 Model 4 uses LDOS as its operating system but you can buy CP/M for this microcomputer from an independent software manufacturer. Purchasing an educational application for the TRS-80 Model 4 does not guarantee that the program will operate under LDOS. It may require CP/M in order to operate. This can get even more complicated with microcomputers like the IBM PC which can use PC DOS, MS-DOS, CP/M and more. The programming language of the application is also important because this language might have to be invoked before the application can be run. Those applications written in machine language or assembly language are usually automatically executed by their system software. However, there are several educational packages that require one to get into BASIC before the application can be run.

In considering both the hardware and the software issue in educational software evaluation, the main problem arises when either the advertising is not specific enough to allow the purchaser to determine whether or not the application package will operate properly on his microcomputer system, or the purchaser does not understand the information that is being made available in the advertising. This last issue serves as an introduction to the people consideration in software evaluation. There are many old adages that apply here about people, but it will probably be most helpful if you remember just two: "If all else fails, READ THE INSTRUCTIONS." and "You can please all of the people some of the time, and some of the people all of the time, but you can't please all of the people all of the time." Software manuafacturers realize this last point all too well because they actually target their software to provide an "80 percent solution." To complicate matters, there are about five different sources for educational software, and each has a different purpose for producing its software. The sources for educational software are Hardware Manufacturers, Textbook Publishers, Educational User Groups, Independent Software Houses, and Classroom Teachers. Thus, it is a foregone conclusion that all teachers will not evaluate educational software packages in the same manner, so the only way for software manufacturers to produce a successful educational product is to provide a package that does MOST of what ALL the teachers would want it to do.

Educational software evaluation does not have a long history nor does it have a proven record of quality control. In the March 1981 issue of The Mathematics Teacher (publication of The National Council of Teachers of Mathematics -- NCTM), the headline on the editorial page read, "Software Reviews Are Coming." The first such review appeared in the next issue, but it was not until October 1981 that a steady stream of software reviews began appearing in each subsequent issue. During that same period, the National Science Foundation funded a project for the dissemination of software evaluations by the Northwest Regional Educational Laboratory. This project was called MicroSIFT. Both NCTM and MicroSIFT have published evaluator's guides for educational software evaluation. The NCTM guidelines are meant to assist teachers in conducting their own evaluations. The MicroSIFT guidelines are much more formal and rely on experts in the particular fields.

In general, there are three basic stages in an evaluation: the classification stage, the descriptions stage, and the observation stage. In the classification stage, you look at the software package to determine the nature of the educational use (administrative, CAI, or programming). Furthermore, if it is CAI, then you must determine what type of CAI it is (drill and practice, tutorial, etc.). Finally in this stage, you must assure yourself that the educational package operates correctly without any problems on the appropriate microcomputer system and that the manufacturer has given sufficent documentation for this proper operation. In the description stage, you identify the factual information necessary for someone to use this software. You identify the manufacturer's name and address, a HOT-line telephone number if available, the grade level of the software (i.e., this is appropriate for visually impaired children in third grade), the appropriateness of the subject matter, the mode of instruction (i.e., drill and practice, tutorial, etc.), the required hardware configfuration, the required system software, the instructional objectives, and any prerequistes that might be necessary. In the observation stage, the software should be used by someone in the intended audience (i.e. a visually impaired child in the third grade) and observational notes taken. A checklist for each of the different types of CAI is given as Figure 2. However, general characteristics that you should be aware of are as follows: (1) User interfaces require little effort (few keystrokes or simple screen navigation); (2) User does not have to memorize commands, functions, or processes (You do not want the student to have to refer to the manual continually in order to use a drill and practice software package); (3) The software is not frustrating to use because it is too slow or dull; (4) The software prevents the user from accidental problems, that is, deletions, quick exits, and other functions that may erase data or files (Any deletions or exits from the program should be double checked by the software before the function is executed.).

Finally, you should realize the pedagogical problems of evaluating educational software by asking yourself certain questions. Is the computer useful in the educational process which this software attempts to teach? Is the software intrinsically fun to use so that the students will want to use it over and over again? Does the software enable the users to practice what they are learning? Does it allow for variation and experimentation so that learning becomes internalized? Can you or a student modify the software? Would you purchase this software for your own children?

The Swing

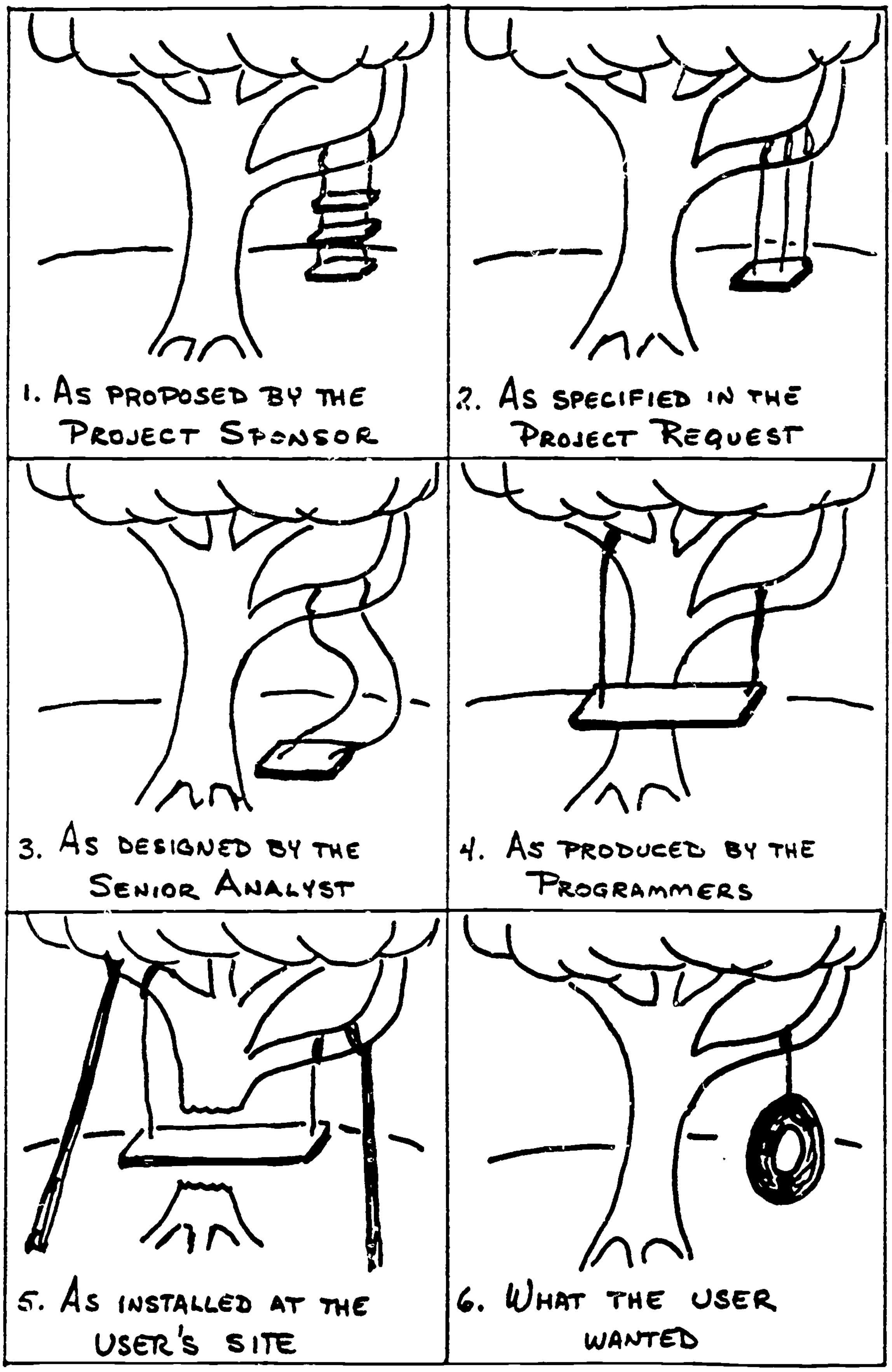

1. As proposed by the Project Sponsor
2. As specified in the Project Request
3. As designed by the Senior Analyst
4. As produced by the Programmers
5. As installed at the user's site
6. What the user wanted

Figure 1

Evaluating CAI Software

Drill and Practice:
1. Identifies prerequisite skills or required knowledge through pretesting.
2. Reinforces skills previously taught.
3. Varies exercises by generating data randomly so as to avoid user boredom or memorization.
4. Branches to easier or more difficult tasks based upon the user's responses.
5. Provides explanation of process being drilled and provides correct answers for incorrect user responses.

Tutorial:
1. Bears full task of instruction, incorporating user responses.
2. Presents initial, basic instruction to teach a skill, concept, or process.
3. Assumes that the user has little or no prior instruction in the subject matter being taught.
4. Presents information, directions, and processes in clear, sequential steps.

Testing:
1. Allows for the collection, storage, retrieval, and report of student test scores and class records.
2. Provides randomly generated tests from a pool of test questions and with each test provides the answer key.
3. If the testing is done on the computer, the software should also provide monitoring of the time for each student's response, diagnoses of the student's responses and progress, and suggested follow-up instruction.

Simulation:
1. Replicates the vital aspect of a real situation.
2. Provides for sufficient user involvement to make experience meaningful.
3. Requires the user to do problem solving and decision making by suggesting "what if" possibilities.
4. Presents activities too difficult, dangerous, expensive, or inconvenient for users to experience firsthand.

Dialog:
1. Provides for problem-solving experience which goes beyond either simple or typical word problem application.
2. Requires the user to apply accepted principles to determine responses.
3. Provides the user an opportunity to create or analyze variations of the problem based upon input from the user.
4. Includes an explanation or graphic illustration of the resultant situation from the responses made in the solving of the problem.
5. Develops an appreciation and understanding of algorithms.

Figure 2