INFORMATION TO USERS

This reproduction was made from a copy of a document sent to us for microfilming. While the most advanced technology has been used to photograph and reproduce this document, the quality of the reproduction is heavily dependent upon the quality of the material submitted.

The following explanation of techniques is provided to help clarify markings or notations which may appear on this reproduction.

1. The sign or "target" for pages apparently lacking from the document photographed is "Missing Page(s)". If it was possible to obtain the missing page(s) or section, they are spliced into the film along with adjacent pages. This may have necessitated cutting through an image and duplicating adjacent pages to assure complete continuity.

2. When an image on the film is obliterated with a round black mark, it is an indication of either blurred copy because of movement during exposure, duplicate copy, or copyrighted materials that should not have been filmed. For blurred pages, a good image of the page can be found in the adjacent frame. If copyrighted materials were deleted, a target note will appear listing the pages in the adjacent frame.

3. When a map, drawing or chart, etc., is part of the material being photographed, a definite method of "sectioning" the material has been followed. It is customary to begin filming at the upper left hand corner of a large sheet and to continue from left to right in equal sections with small overlaps. If necessary, sectioning is continued again—beginning below the first row and continuing on until complete.

4. For illustrations that cannot be satisfactorily reproduced by xerographic means, photographic prints can be purchased at additional cost and inserted into your xerographic copy. These prints are available upon request from the Dissertations Customer Services Department.

5. Some pages in any document may have indistinct print. In all cases the best available copy has been filmed.

University Microfilms International
300 N. Zeeb Road
Ann Arbor, MI 48106

Calamari, Mary Diane Bodin

A COMPARISON OF TWO METHODS OF TEACHING COMPUTER PROGRAMMING TO SECONDARY MATHEMATICS STUDENTS

The Louisiana State University and Agricultural and Mechanical Col. PH.D. 1983

A COMPARISON OF TWO METHODS OF TEACHING COMPUTER PROGRAMMING TO SECONDARY MATHEMATICS STUDENTS

A Dissertation

Submitted to the Graduate Faculty of the
Louisiana State University and
Agricultural and Mechanical College
in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

in

The Interdepartmental Program in Education

by
Mary Diane Bodin Calamari
B. S., Louisiana State University, 1961
M. Ed., The University of Southwestern Louisiana, 1969
Ed. S., The University of Southwestern Louisiana, 1978
December 1983

ACKNOWLEDGMENTS

The writer gratefully acknowledges the assistance of Dr. B. F. Beeson, major professor, for his advice, confidence, and encouragement during this study. Appreciation is directed to other members of the doctoral committee for their guidance: Dr. Stephen M. Buco, Dr. Donald H. Kraft, Dr. Richard Lomax, Dr. Anthony W. Romano, and Dr. John A. Hildebrant, minor professor.

Additional expressions of gratitude are due to school officials in St. Mary Parish, to the principals of Berwick High School, Franklin Senior High School, and Patterson High School, to those mathematics teachers who allowed me to teach their students programming, and to the students who participated in the study. Appreciation is extended to the instructor and students in the computer science class at University High School in Baton Rouge, Louisiana, for field testing the evaluation instrument. The writer acknowledges the aid of those people who secured a computer for use in the study.

Gratitude, appreciation, and thanks are given to family and friends who provided encouragement whenever this was needed.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

This study was formulated in order to gain information about the effectiveness of two methods of instructing secondary mathematics students in introductory computer programming. Investigation of programming instruction in a reduced period of time was a major component of the study for those educators who choose to teach programming as a portion of a mathematics course must do so under time constraints.

The research was conducted in three public secondary schools in St. Mary Parish in Louisiana. The subjects were enrolled in an algebra II or a geometry class during the spring semester of the 1982-83 academic year. An experimental-control group, pretest-posttest design was employed in the study. The experimental group was composed of sixty-nine students, and the control group consisted of sixty-two students.

The purpose of this experiment was to investigate the effect of two methods of teaching computer programming using the BASIC language. Subjects in the experimental group received instruction in the turtle graphics component of Logo, a simple language designed for a child's initial computer experiences, and they designed two programs using this language. The Logo phase of the treatment accounted for approximately 25 percent of the total treatment time.

For the remainder of the treatment period of nine hours the experimental group received instruction in BASIC, and the students created programs in that language. The control group was taught to design programs in BASIC only so that all of the programs created by the subjects in the control group were in BASIC. Treatment for both groups was conducted by the researcher.

A pretest on computer programming in BASIC was administered to the subjects on the first day of the experiment, and the participants completed a posttest immediately after the treatment. The evaluation instrument was designed by the researcher, field tested in a secondary school, and validated by a panel of experts.

Analysis of covariance was performed on the scores obtained on the BASIC programming posttest using the pretest score as covariate. The students who had received instruction in BASIC alone exhibited significantly higher achievement than those taught Logo and BASIC. No significant difference was found in the scores of samples of algebra II students and geometry students nor in samples of males and females who received instruction in programming by each of the two methods of study.

CHAPTER ONE

Introduction

Computer technology influences the everyday lives of many people, and there is no evidence that its impact will lessen in the future. Computer applications are utilized across a broad spectrum of occupations in the United States and throughout the world. Harold G. Shane (1982) reports that information workers constituted about half of the total labor force in 1980. These workers include handlers of information and dispersers of communications. The percentage of people employed in information occupations has steadily increased during the last decade. Since secondary students are the labor force of the future, many schools are purchasing inexpensive microcomputers in order to provide computer experiences for their students. Thus, the influence of the computer is directly involved in the education of children. Although there are several educational uses for computers including computer assisted instruction, administrative applications, and computer programming, the one way by which a person can control the electronic device is by learning how to program. Computer programming, then, becomes a component of the secondary school curriculum.

In many secondary schools students are given the

opportunity of receiving instruction in introductory computer programming. Often it is the mathematics teacher who interweaves programming and mathematics because computer applications in this field are obvious. The task is accomplished when the teacher emphasizes problem solving as an essential component of mathematics and utilizes the computer as a problem solving tool. The computer is an ideal device for teaching mathematical thinking through problem solving (e.g., Norris, 1981; Piele, 1982; Wold, 1983) and thus the mathematics classroom contains an instructional tool of major importance. Use of the computer in this manner requires that students design programs to serve as the solution to their problems. Thus, the mathematics teacher must also be a teacher of fundamental computer programming.

The secondary mathematics instructor must select a method of teaching introductory programming in a reduced period of time because computer work is only a portion of the content of the mathematics course. Since BASIC is the most popular computer language used today (Wold, 1983), instruction is most likely to be in that language. Even though these students may be rather mature, they are novices at programming and must begin at the lowest level regardless of their chronological age.

Several K-12 computer literacy plans note the progression found in the formation of computer competency. One of these programs teaches Logo as an elementary language in

lower grades followed by instruction in BASIC (Fisher, 1983), while a secondary school in Massachusetts uses Logo as an introductory language in the regular computer science class (Watt, 1982b). The traditional method of teaching BASIC programming involves instructing the students in that language exclusively. This study investigates these two methods of teaching introductory computer programming to secondary mathematics students.

Two programming languages, BASIC and Logo, are designed for initial instruction in programming because of their simplistic nature. BASIC was originally developed at Dartmouth College by John Kemeny and Thomas Kurtz in the middle 1960s. The language was adopted by several time-sharing systems which compelled many users to become familiar with it. The greatest impetus to the use of BASIC came in the mid-1970s when the microcomputer was invented. Nearly all microcomputers are equipped with a BASIC interpreter so that BASIC is their standard host language. The simplicity and convenience of BASIC has been a significant factor in the popularity of the microcomputer (Gottfried, 1982). Most students in introductory programming classes at the secondary level are taught BASIC.

An intensive course in BASIC programming was offered in a college for a shortened period of time. Glassboro State College in New Jersey offered a college-level, twelve-day BASIC programming course between semesters in 1979 to avoid

overcrowding in the regular class (Masart, 1981-82). Students in this shortened class showed achievement in programming equal to or greater than that of students in the regular semester class. Thus, students can learn to program in a compressed period of time.

A simple, special purpose language dedicated to the education of children was developed by Seymour Papert at Massachusetts Institute of Technology. Logo was originally designed for lower elementary children, but it has been used as a teaching tool throughout elementary school and even with older children and adults (du Boulay and Howe, 1981; Solomon, 1975; Watt, 1982b). A major component of Logo, called "turtle graphics", consists of a few commands that produce graphical output when a program containing the commands is executed. The output is specified by stating translational and rotational vectors. Although the language was designed for use with young children, Logo could be used profitably in secondary schools because of its relation to polar coordinates and vectors.

The developers of Logo performed a comprehensive study of the effect of learning the language on both programming and mathematics achievement of sixth graders at the Lincoln School in Brookline, Massachusetts, during the 1977-78 school session (Papert, Watt, di Sessa, and Wier, 1979). Every sixth grader was taught Logo, and, in addition, sixteen students were singled out for more extensive study.

Six of these students scored below average on achievement tests and were rated as below average by teachers; six were labeled "above average"; the remaining students were average. The students received four hours of Logo training per week for from four to eight weeks.

The objectives for the computer programming phase of the experiment are relevant for they demonstrate the expectations of the designers of Logo who feel that students should achieve the following goals:

> (A1) The idea of computer programming in a formal language; its syntax, effect, and associated error messages; the LOGO commands FORWARD, RIGHT, etc. and the arithmetic operations;
> (A2) The idea of sequential procedure and the ability to translate an informally defined plan into a working program; the LOGO commands TO, END;
> (A3) The use of subprocedures and superprocedures;
> (A4) Editing and debugging; LOGO commands EDIT, PO, POTS, etc.;
> (B1) Control of continuing processes with loops and/or recursions;
> (B2) Use of variables;
> (B3) Conditionals and stop rules; and
> (B4) Writing interactive programs. (Papert et al., 1979: 11)

Only two of sixteen students failed to meet the objectives listed for minimum programming proficiency, those which are indicated as A1 through A4. Both of these students were in the lower quartile in school performance (Papert et al., 1979: 20). This provided evidence that Logo can be used to teach the concept of a computer program and initial computer programming skills.

Children seem to enjoy Logo, and many users of the

language have consolidated to exchange ideas. One user group is composed of young people involved in an adult-like organization. This club is described in Educational Computer Magazine (1982). The Young Peoples' LOGO Association was formed in Richardson, Texas, by a man and his learning disabled son. The original purpose of the group was to serve as a central repository for Logo software; secondary goals include promoting the educational use of computers and training the handicapped. This association has developed an apprentice program for secondary students who desire a career in computer science where they are taught Logo as a supplementary programming language (Muller and Muller, 1982). Larry, a learning-disabled eighth grader, is able to program a computer and enjoys doing it. Computer programming, then, is not restricted to the intellectually elite.

The BASIC programming achievement of students who learn Logo before learning BASIC is compared to the achievement in BASIC programming of students who are taught the BASIC language exclusively. Since purchasing the Logo language option for a microcomputer is an added expense, the effectiveness of its use in programming instruction at the secondary level should be investigated.

Statement of the Problem

This study compared the achievement in BASIC computer

programming of secondary mathematics students who were taught both BASIC and Logo to those instructed only in BASIC. The experimental group designed programs in the Logo language for a specified period of time before creating programs using the BASIC language. The control group programmed in BASIC for the entire treatment period. The measured criterion was the score achieved on a BASIC programming test prepared by the researcher.

The following hypotheses were tested:

1. There is no significant difference in achievement in BASIC computer programming between secondary mathematics students who are taught programming using BASIC after they learned Logo and those who are taught programming using only BASIC.

2. There is no significant difference in computer programming achievement between secondary geometry students and secondary algebra II students.

3. There is no significant interaction between type of mathematics course (algebra vs. geometry) and type of treatment (BASIC with Logo vs. BASIC alone).

Conceptualizations

Logo or LOGO is a simple, special-purpose, high level programming language with relatively few commands that was designed to be an initial programming language for children. Turtle graphics commands in Logo consist of words denoting

direction and numerals to designate angles or amount of lateral motion. Logo produces primarily graphical output.

BASIC, Beginner's All-Purpose Symbolic Instruction Code, is a general purpose, high level, initial programming language that includes commands for numeric, character, and graphical output. Instructions in BASIC consist of algebraic formulas or alphanumeric characters used in association with certain English keywords and symbols for variables. BASIC programming can be used in a variety of applications in diverse fields.

A computer program is a step-by-step plan for solving a problem written in a language that communicates with the computer. Creating a program is called computer programming.

The term computer, in this study, refers to a microcomputer. A microcomputer has a relatively small amount of accessible memory, often 64K or less, and is constructed so that the central processing unit is an integrated circuit on a microprocessor chip. It is a single user device.

The researcher considers a secondary student to be any student enrolled in a geometry class or an algebra II class in a St. Mary Parish public school regardless of the grade placement of the student. The public schools in St. Mary Parish that offer classes in geometry or algebra II are Berwick High School, Centerville High School, Franklin Senior High School, Morgan City High School, and Patterson High School.

CHAPTER TWO

Review of Related Literature

Although there are over 17,000,000 microcomputers in United States schools (Lias, 1982), there is little agreement about who will teach computer skills in those schools. This fact is obvious when one reviews the results of a survey sponsored by North Texas State University in which the chief education official of each of the 50 states reported on the status of computer science certification in his/her state (Taylor and Poirot, 1983). Responses to the survey indicate that computer science certification as an entity is available in only four states. Those states that reported certification as part of another discipline listed the areas of mathematics, business, science, and industrial arts as those which could be combined with computer science for certification purposes. In the states that had no requirements at all, mathematics was the subject area from which most of the computer science teachers were recruited.

Since so much computer science is taught by mathematics teachers, the subject frequently has a mathematical flavor. In addition, many teachers are using the computer as a problem solving tool in the mathematics classroom which involves teaching students how to program and, most likely,

how to program in an abbreviated period of time. Investigation of the effects of two methods of teaching programming in a reduced time frame on student achievement was the major research question of this study.

Computers and Programming

Several well-known national organizations have recognized the importance of computer education by publishing statements that reflect its significance. In 1972 the Conference Board of the Mathematical Sciences Committee recommended the institution of a computer literacy course for all junior high school students followed by a BASIC programming course that emphasizes problem solving (Hunter, Kastner, Rubin, and Seidel, 1975: 276). This recommendation was conceived before the development of microcomputer hardware so few schools were able to act on the statement. The same viewpoint was found in a 1980 publication of the National Council of Teachers of Mathematics which advocated that "mathematics programs must take full advantage of calculators and computers at all grade levels" (National Council of Teachers of Mathematics, 1980: 8). Some of the means recommended to accomplish this goal include the integration of calculators and computers into core mathematics courses, the development of a computer literacy course for all students to familiarize them with the role of this electronic device and its impact, and the formulation of a

secondary computer science course of sufficient depth and breadth to provide students with a background for advanced work in computer science. Similar recommendations were prepared by the National Council of Supervisors of Mathematics (National Council of Teachers of Mathematics, 1978).

Each of these sets of recommendations was developed by a national organization which serves mathematics education. In fact, the history of the computer is closely tied to the field of mathematics. Many of the pioneers in the development of computers (e.g., Blaise Pascal, Gottfried Leibnitz, Charles Babbage, Ada Countess of Lovelace, Howard Aiken, John von Neumann, Stanislaw Ulam, John Kemeny, Patrick Suppes, and Seymour Papert) were mathematicians (Piele, 1982: 132). The areas of mathematics and computer science have been closely allied from the beginning.

A report from the National Council on Excellence in Education, A Nation at Risk: The Imperative for Educational Reform, stated that all United States secondary schools should emphasize five basic skills, English, mathematics, science, social studies, and computer science (Morning Advocate, 1983: 1) in order to improve the quality of education in American schools. This committee recommended requiring one-half unit of computer science for all students.

Thoughts of computer science as a basic skill have been expressed by other individuals prior to the report of the National Committee on Excellence in Education. In 1981,

Arthur Luehrmann, then the associate director of the Lawrence Hall of Science at the University of California at Berkeley and outspoken advocate of teaching computer programming to all students, expressed the following argument for including computing in the school curriculum:

> Computing belongs as a regular school subject for the same reason that reading, writing, and mathematics are there. Each one gives the student a basic intellectual tool with wide areas of application. Each one gives the student a distinctive means of thinking about and representing a problem, of writing his or her thoughts down, of studying and criticizing the thoughts of others, and of rethinking and revising ideas, whether they are embodied in a paragraph of English, a set of mathematical equations, or a computer program. Students need practice and instruction in all these basic modes of expressing and communicating ideas. (Luehrmann, 1981: 686)

In an earlier report Luehrmann predicted that the "ability to use computers will be judged a basic skill in the near future" (Luehrmann, 1980d: 154), but he bemoaned the lack of a recognized curriculum for teaching computer skills and the absence of a method to measure competency in computing.

Another educator, Eileen K. Gress, identifies computer literacy as a basic skill and reports that she is certain that some form of computer education will become a part of the school curriculum (Gress, 1981). In an often-quoted story, Luehrmann (1980c) makes the point that computing is as necessary to a person as is the skill of writing. In order to assert the significance of every student possessing computer skills, Luehrmann describes the unique

content of a computer class by pointing out that:

> The next step will be the creation of new courses specifically to teach the ordinary students ... how to use computers for a variety of tasks, both numeric and non-numeric. These 'basic computing skills' courses will teach about algorithms, writing and debugging programs, running simulation models, editing text, and drawing graphs and pictures. (Luehrmann, 1980a: 145)

Some of the skills acquired in computer classes can be presented easily in conjunction with the electronic medium, but the same skills are presented with much difficulty elsewhere in the curriculum.

Two school officials in New York envision the day when each American child will own at least one personal computer (Sobol and Taylor, 1980: 155), and Judith Hopper, a junior high school teacher in Colorado, sees computer education as necessary for the future citizen so that he/she will have "better career opportunities and less career shock" (Hopper, 1980: 62). Several educators consider a person to be computer literate only if his/her education contains some programming (e.g., Reitz, 1983; Watt, 1982c). Gay Reitz, an elementary school teacher in New York, views programming as the only method by which children can understand and use computers in a realistic manner (Reitz, 1983: 26). Daniel Watt (1982a: 45) describes the advent of the microcomputer as an impetus for offering computer programming to students of all ages.

Donald Norris (1981), a college professor from Ohio

University, proposed that college preparatory mathematics be altered by replacing plane geometry with computer programming. This would mean that the core courses in mathematics for the college-bound student would include two years of algebra and one year each of computer programming and a precalculus survey. Electives to this proposed curriculum could include plane geometry, calculus, or a higher level computer class. Some of the benefits accrued from learning to program a computer include practice in logic and precision (Sobol and Taylor, 1980), two of the reasons often expressed for studying plane geometry. Since programming allows students access to the computer and since use of the machine motivates the student, Atchison (1978) asks that teachers structure courses so that students begin actual work on the computer within the first few days of class.

There are many other benefits of teaching computer programming. Programming allows the operator to control the machine (Luehrmann, 1980c; Watt, 1982c); this feeling of control over technology gives a student self-confidence (Watt, 1982c). Other aspects of learning associated with computer programming involve the development of a set of problem solving strategies, gaining an understanding of the subject matter of the program, and studying about abstract machines (Abbott, 1979; Sobol and Taylor, 1980; Watt, 1982c). In a statement presented to the House Committee on

Science and Technology, Arthur Luehrmann described the skills acquired when one learns how to use the computer as "programming, structural thinking, and critical evaluation of computer applications... skills that are presently lacking in the public..." (Luehrmann, 1980b: 137). Development of problem solving skills through programming is identified by Eileen Gress, a mathematics teacher in New York, as the area of greatest computer impact (Gress, 1981: 42).

Beverly Hunter (1982) conducted a review of computer education historically from 1949. Highlights of the history include the organization of the first computer class in a secondary school in 1961, the invention of the BASIC programming language in 1963, and the development of the personal computer in 1975. In 1963 only 13 percent of the secondary schools in the United States used computers in some form of instruction; by 1973 the figure had risen to 26.7 percent. The relatively short history of computer education accounts, in part, for the lack of a standard computer science curriculum and for the lack of agreement concerning the content of computer skills courses. Robert D. Gilberts, Dean of the College of Education at the University of Oregon, describes the "fourth or fifth generation of computers in the 'second computer revolution' for the world" as the "first computer revolution for schools" (Gilberts, 1982: 4). He suggests that the absence of solidified usage of computers in education is due to the

high cost of hardware and the rapid changes in the technology of the electronic machine.

A survey of students enrolled in a computer education program taught at seventy-four secondary schools in Yugoslavia revealed that half of the student respondents thought that all students should take computer science classes; 33 percent replied that computer science should be an available elective (Bratho, Rajkovic, and Roblek, 1975). Thirty-four percent of their teachers responded that computer science should be a required course; 48 percent of them thought that it should be an elective. Whether computer science should be a necessary component of general education yielded a more pronounced trend where 87 percent of the students and 91 percent of the teachers viewed computer science as an essential part of general education.

Daniel Watt (1982a) points out that many colleges will soon require computer programming for admission as Harvard does now. Since preparation for college is one of the goals of secondary education, programming will be emphasized more in pre-college education from necessity if not from perceived benefits. A survey of teachers in Minnesota revealed that only 32 percent of Minnesota teachers thought that all students should be able to write a computer program (Hansen, Klassen, Anderson, and Johnson, 1981: 469). Results of the same survey indicated that mathematics teachers in Minnesota use computers in their classes to

teach terminal operation, programming, and student problem solving, in that order. Other surveys on computer usage report that computers are utilized in Montana to teach mathematics, computer education, and programming (Dolan, 1982: 58), and that throughout the nation the most popular educational uses of computers are in mathematics as reported by 79 percent of the schools surveyed and in computer science as indicated by 53 percent of them (EL News, 1982: 12). The latest National Assessment of Educational Progress reports that only 14 percent of 13-year old youths and 12 percent of 17-year olds had any computer instruction in their mathematics class while only 12 percent of junior high respondents and 25 percent of the older students had access to computers in their schools (Carpenter, Corbitt, Kepner, Lindquist, and Reys, 1980: 671). Thus, the perception of the significance of education in the use of the computer has not been realized.

Problem Solving

Frequently an educational belief is characterized by a slogan. In the 1980 yearbook of the National Council of Teachers of Mathematics the following statement was made:

> In the late fifties we had the 'revolution in mathematics'; in the sixties we had 'modern math'; in the seventies it was 'back to basics.' As we enter the eighties, the theme appears to be 'problem solving.' (Krulik, 1980: xiv)

If this prediction proves to be accurate, problem solving

will be a major concern of mathematics educators.

David Moursund (1980: 127) describes the computer as a "universal tool, a new aid to problem solving." The role of the computer in this significant area was described in a paper presented at the International Foundation for Information Processing TC-3 Working Conference on Informatics and Mathematics in Secondary Schools in 1977 and summarized by the following statement:

> A sound approach to problem solving is perhaps one of the most important things that can be taught at any level. It's quite natural to develop this in computer science and, in particular, in the various levels of programming. This is certainly an area where mathematics and computer science can work cooperatively in developing a student in his problem solving ability. (Atchison, 1978: 28)

Donald Norris expresses the same idea when he states that "it requires a tremendous amount of problem-solving ability to make a computer solve a problem" (Norris, 1981: 26). Eileen Gress (1981) points out that the process of developing a computer program encourages divergent thinking and creativity, two skills needed for successful problem solving.

After noting that teaching problem solving skills is one of an educator's most difficult tasks, Daniel Watt (1982c: 130) describes computer programming as an "excellent means for developing problem-solving strategies." Watt refers to problem solving techniques as espoused by George Polya and others (e.g., Hughes, 1976; Newell and

Simon, 1979; Pereira-Mendoza, 1979; Polya, 1957; Polya, 1962; Polya, 1965; Robison, Tuckle, and Brison, 1972; Rubinstein, 1975; Wicklegren, 1974). Some of the specific heuristic skills (see Appendix E for a list of Polya's heuristic stretegies) which are practiced when one writes a program are subdividing the problem into parts, creative intuition (i.e., because a program is seldom perfect when it is first written), and debugging (i.e., testing, comparing to expected results, revision, which correspond to Polya's "looking back strategies"). Since the computer provides immediate feedback, the learner knows without being told if a given plan for solving a problem needs revision or not. Thus, computer programming is a vehicle for teaching problem solving because creating a computer program, testing it, and modifying it requires the same approach and strategies as does heuristic problem solving.

The link between computer programming and problem solving is the algorithm. According to Horn and Poirot (1981: 187), an algorithm is "a procedure for organizing a logical series of steps to solving a problem. Logic is a problem-solving term that refers to patterns for doing things in a specific order because it is reasonable to do them that way." Development of the algorithm is the most important and most difficult step in solving a problem. The task of the programmer, as described by Terry Winogard, is "to design an algorithm (or a class of computations) for

carrying out a task and to write it down as a complete and precise set of instructions for a computer to follow" (Winogard, 1979: 391). The algorithm which precedes the writing of a computer program is that same concept that is described by Polya as "devising a plan."

Arthur Luehrmann (1980d: 154) states that a person who can devise an algorithm and communicate it to the computer is the possessor of "a new intellectual tool of basic importance to virtually every subject matter now being taught." Such a person has a distict educational advantage over those who cannot use a computer in this constructive, analytical mode." In a 1974 article describing the relationship between mathematics and computer science, Donald E. Knuth considers the algorithm as the most significant discovery of the computer age and relates that "an algorithmic point of view is a useful way to organize knowledge in general" (Knuth, 1974: 323). He also characterizes the knowledge acquired from programming in the following manner:

> A person well-trained in computer science knows how to deal with algorithms; how to construct them, manipulate them, understand them, analyze them. This knowledge prepares him for much more than writing good computer programs; it is a general-purpose mental tool which will be a definite aid to his understanding of other subjects. (Knuth, 1974: 326)

The algorithmic approach as manifested in creating computer programs becomes an important benefit received by those students who learn computer science.

Referring to his twelve years of experience in computer education, Luehrmann (1980b: 139) states that a student who designs a computer program to solve a problem "nearly always has a better understanding of the problem than he or she got from a purely verbal or a purely mathematical description... there is an intellectual dimension to computer problem solving." Other educators indicate the significance of computers in problem solving and their place in the total school setting. Piele (1982: 133) speculates that "mathematical problem solving - constructing algorithms and running them on a computer - will become an accepted part of the mathematics curriculum." Soloway, Lochhead, and Clement (1981) advocate the inclusion of computer programming in the mathematics curriculum or the complete integration of algebra and programming. Several educators see the computer as a viable tool for teaching that most important and most difficult area of mathematics - problem solving.

Computers and Mathematics

As additional evidence of the close association between mathematics and computer studies, the 1977-78 National Assessment of Educational Progress examination in mathematics included four questions that tested programming skills (Carpenter, Corbitt, Kepner, Lindquist, and Reys, 1981). Thus, those who devised the examination realized that

computer programming is a probable component of the mathematics curriculum by including some computing items on the mathematics assessment.

A critic of the manner in which computers are used in mathematics classes, David Moursund, chairman of the Association for Computing Machinery Elementary and Secondary Schools Subcommittee and editor of The Computing Teacher, notes that computers are merely an "add-on topic" because little change has occurred in the content of mathematics classes because of computers. This same situation is found in both business and science classes, also. He advocates that "... content and coursework for every discipline needs to be rethought and redone in the light of computers and their capabilities" (Moursund, 1980: 128). In Mathematics Teaching Maurice Hart (1982) reports on the Nottingham Programming in Mathematics Project conducted in England in which children learn BASIC programming before they study algebra so that they can use the concept of a variable as a storage location. In this project the concept of a variable has been altered from its traditional view as an element of a domain or replacement set.

An experiment performed at the University of Massachusetts at Amherst relating computer programming to solving algebraic word problems is reported by Soloway, Lochhead, and Clement (1982). A computer science class of college freshmen and sophomores was divided in half for the purpose

of presenting a problem in different formats. Each group received a problem having the same content, but for one group the problem was presented as a computer program to be created whereas the other half of the same class received the problem in verbal algebraic terms. Sixty-nine percent of the 52 subjects correctly wrote a BASIC program to solve the problem, but only 46 percent correctly solved it algebraically. The difference in performance is significant at the 0.05 level. This is a fruitful field for exploration since "there is no area in algebra which causes students as much difficulty as word problems (Johnson, 1976: i).

A second experiment conducted by Soloway, Lochhead, and Clement (1982) involves the interpretation of a word problem. The evaluated questions were embedded in a test taken by freshman engineering students. For each of two questions students were asked to read and explain an algebraic formula In problem 1 the formula was presented as a stand-alone entity with each variable identified (the traditional mathematical way), but in problem 2 the formula was part of a computer program with the variables identified in the program specifications. The formulas used in each problem had the same format (e.g., A = 7S and K = I*2) even though different subject matter was invoked. The researchers were interested in those students who missed one of the problems thus illustrating a difference in the two approaches. Five subjects responded correctly to problem 1 only, but 18

students correctly answered problem 2 only. This difference is significant at the 0.005 level. These results are important to mathematics teachers because they show promise for improving instruction in problem solving.

The researchers mentioned several reasons for improved problem solving in a programming environment (Soloway, Lochhead, and Clement, 1982: 179). The explanations include consideration of the requirements imposed upon the user by the programming language such as definite, unambiguous, explicit syntax and semantics and of the programmer's approach to the problem where an equation is viewed as the transformation of an input to an output. Requirements for effective programming such as decomposing the problem specifications into small steps in order to design a program to solve it and analyzing both the problem and its solution when debugging a program are additional practices that enhance problem solving with computer programming.

Douglas Davis (1980), a physics professor at Eastern Illinois University, reports on the use of computer programming and numerical analysis to teach a course in traditional classical mechanics. During the first week of the class the students were taught enough BASIC to write, run, and debug simple programs. Then they used the computer to solve the physics problems that composed the subject matter of the class. Douglas points out that students found computer work to be both challenging and enjoyable; students in this

computer-based class gained a deeper understanding of the concepts illustrated in the problems.

At the University of Minnesota computer programming is used to teach number theory. Sabra S. Anderson (1982: 91) describes this instruction as discovery teaching in the spirit of Polya. After each topic is introduced, a computer project on the topic is completed. Then the topic is discussed followed by synthesis, proofs, and unsolved problems. Programs such as the determination whether a given number is prime or composite are used in the class. A class prerequisite is a knowledge of BASIC or FORTRAN programming language.

If a secondary mathematics teacher wishes to implement computer-aided problem solving, he/she must provide the student with instruction in programming because most high school courses do not have an extensive set of prerequisites. This involves a portion of the mathematics class that is not directly devoted to instruction in mathematics. Since all microcomputers are equipped with a BASIC interpreter, this instruction will most likely be in BASIC. Generally the programming unit is taught for a specified, shortened period of time. When choosing the content of the programming unit, the teacher should consider the statement by Herb Nickels (1980), the instructional computing coordinator at California State College in San Bernadino, that ten percent of the legal commands of a programming language

constitute ninety percent of the code in that language. He lists the essential commands for BASIC as "PRINT, INPUT, LET, GOTO, IF...THEN, DATA, READ, FOR, NEXT, DIM, GOSUB, RETURN, and REM" (Nickels, 1980: 158). Any unit of computer programming in BASIC should contain at least this set of commands.

Initial Informal Programming

Teachers are always searching for meaningful initial experiences on a topic of study because one's first impression of a subject is often a lasting one. In the preface of his textbook on informal programming for college students, Richard E. Pattis (1981: vii) identifies the first few weeks of a student's programming instruction as most significant to that student's perception of the subject because at that time he/she can see the overall aesthetics of programming and is flexible enough to react favorably to new ideas.

Pattis' (1981) solution to this pre-higher language stage in programming is to use a simple language with few commands to move Karel the Robot across a grid that is superimposed on the computer screen. The student is given a problem in which the robot is to move to a specified location, pick up or put down a beeper, and repeat some movement or halt; five commands constitute the instruction set used to accomplish this. The student's task is to create

a program using the command set to cause Karel to move as specified in the problem. This informal programming language is used at Stanford University for four days at the beginning of an introductory programming class in Pascal. There is much similarity between Karel and the turtle graphics of the Logo language (Haney, 1981), and in the preface of his book, Pattis thanks Seymour Papert and the Logo research team for their influence on his thoughts (Pattis, 1981: xi).

An advantage of using a simplified informal language before studying a higher level language is that of immediate access to the computer for problem solving due to the ease with which the language is learned. The student is able to program a computer with confidence because the fundamentals of programming are taught in a simplified fashion yielding a high probability of success in the initial programming experience. The concrete notion of moving a robot across a screen is a realistic beginning point for the novice. The study of Karel also provides beginners with a programming mind set and an appropriate vocabulary to utilize in later study (Pattis, 1981).

Antfarm is a structured programming game that is used to introduce a child to the concept of structured programming before he/she learns a higher level language with the hope that the child will transfer the knowledge to the higher language (La France, 1980). The command set for

Antfarm has only five imperatives which can be used to modify existing programs. This language, like Logo, has procedures that are easily remembered and later combined to compose a larger program.

At the University of Waterloo in Canada Farhad Mavaddat (1981) uses mazes with two instructions (STEP and RIGHT) as a means of initial programming instruction. This format allows study of sequencing, procedures, looping, and conditional statements, all of which are important concepts in program design. Students are presented a maze for which they are to create a program to traverse it; instruction with these mazes lasts from two to six hours. Mavaddat (1981: 49) encourages the use of this type of experience because the student is removing the mystery of the computer at the same time that he/she is learning how to construct programs.

Logo

The Logo language was designed by Seymour Papert and his colleagues at the Massachusetts Institute of Technology as a vehicle for one's initial computer programming experience (Papert, 1980). The language uses a simple command set that can be employed to create sophisticated structured programs and was developed for the beginner of any age (c.f., du Boulay and Howe, 1981; Lough, 1983; Solomon, 1975; Watt, 1982b). Designed by a mathematician, computer

experiences with Logo place the learner in an educational environment. The user is free to explore the programming commands by trying them out and gaining immediate evaluation. Research conducted at Lincoln High School in Brookline, Massachusetts, the Lamplighter School in Dallas, Texas, and the New York public schools has shown that Logo has a significant effect upon the development of problem solving skills, the practice of verifying ideas and theories, and the understanding of mathematical concepts such as those of variables, symmetry, and geometric shapes (Lough, 1983: 50). Lough reports that the name "Logo" was derived from the Greek word "logos" meaning thought or word. This choice for the name of the language from the same root as "logical" provides insight into the intentions of its developers. The simplicity of the language is evident when one examines the turtle graphics facet of Logo which is so elementary that it can be learned through usage in an interactive manner.

Interest in Logo was accelerated recently, and several periodicals have published issues dedicated to Logo and its use. These include the August 1982 issue of BYTE, the November 1982 issue of The Computing Teacher, and the April 1983 issue of Classroom Computing News. The feature article in the March 1983 issue of Electronic Learning was on the use of Logo for teaching programming to beginners. In January 1983, The Computing Teacher began a regular section

called "The Logo Center." In an article in Classroom Computing News, Ricky Carter (1983: 36) listed eight magazines, fourteen magazine articles, six newsletters, eight books, four forthcoming books, and four activity sources as means to obtain information on Logo. Many educational computing conventions have sessions and workshops devoted to teaching with Logo.

Logo has been used successfully to teach mathematically-deficient college students in Scotland certain mathematical concepts (du Boulay and Howe, 1981). Average and below average secondary students improved significantly on a mathematics achievement test after completing a year of Logo programming and a year of programming and mathematics (Howe and Ross, 1981). At Lincoln-Sudbury Regional High School in Massachusetts, Logo serves as an introductory programming language (Watt, 1982b). Logo is also used as an introductory programming language in the K-12 program for developing computer literacy in the Alameda County School District where BASIC is taught after students receive instruction in Logo (Fisher, 1983). Logo, then, can serve a double function by facilitating the acquisition of mathematics and problem solving skills as well as programming skills.

Because of its versatility, simplicity, and power to motivate, Logo might prove to be "the most significant educational software of the decade" (Lough, 1983: 49).

Mathematics teachers should become familiar with this language and its potential for improving mathematics, problem solving, and programming competencies.

The BASIC Language

Beginner's All-Purpose Symbolic Instruction Code or BASIC was developed at Dartmouth College in the early 1960s to instruct students with no previous experience in programming (Andersen, 1982: 52). Two components of the name of the language are significant. The language is intended to be used for introductory programming, and it is a versatile tool that can be used to solve many types of problems. BASIC is the most popular of the more than 150 computer languages in existence today (Wold, 1983).

Many secondary schools offer a full year or a half year of instruction in BASIC computer programming. However, this time schedule cannot be used in computer-assisted mathematics classes. Programming courses of shorter duration must be developed to use in mathematics classes. The Freeport (Illinois) School District offers a nine-week computer literacy course to all seventh graders in which each student learns to write simple BASIC programs (Bangasser, 1983).

An experimental program that is being conducted at Virginia Military Institute supplements the content of a finite mathematics course with computer applications

(Abernathy, Piegari, and Thorsen, 1980). Before entering the class, each student takes four hours of BASIC programming training; the class begins with four or five lectures on BASIC followed by short talks about programming as is necessary. Each student has to complete eight programs on matrices and probability as part of the course requirements. A control group worked the problems without computers. The computer programmers achieved a significantly higher score on the twenty-one common questions from the final test for the course.

Other short courses in computer programming have been used in junior high school, secondary school, and college. At Homestead (Wisconsin) High School, every geometry student is taught a two-week computer literacy unit with the design of a computer program required as a final project (Patton, Ortho, and Hopfensperger, 1981). A three-week computer unit that is taught to students in grades seven through nine devotes week three to the creation of a program that the student can use in one of his/her other classes (Joseph, 1979). The concentrated BASIC course offered between semesters at Glassboro (New Jersey) State College produced programming achievement equivalent to that of students who were enrolled in the regular semester course (Masat, 1981-82).

CHAPTER THREE

Design of the Study

The study was an experimental-control, pretest-posttest design which was conducted using a sample of certain mathematics students enrolled in the public schools of St. Mary Parish in Louisiana. One hundred thirty-one students participated in the study; 60 students were in classes that received the experimental treatment, and 62 were in the control group.

The Sample

The sample consisted of intact classes of geometry and algebra II students that were randomly selected from the total number of classes in these subjects as scheduled in the five secondary schools of St. Mary Parish during the 1982-83 school session. For this school year fifteen sections of geometry and ten of algebra II were scheduled in St. Mary Parish public schools (Jacquet, 1983).

The researcher randomly selected four geometry classes and three algebra II classes to compose the sample. A set of computer-generated random three-digit numerals which designated the school, mathematical discipline, and section number were used to select the classes for the sample. The classes chosen were at Berwick High School (2), Franklin

Senior High School (2), and Patterson High School (3). Each class was randomly assigned to either the experimental or the control group. The researcher obtained permission from the parents or guardians of each student for them to participate in the study. A copy of the letter used to obtain this permission is included in Appendix C. Approximately 300 students were enrolled in geometry and 240 in algebra II throughout the parish; the sample consisted of 62 algebra students and 69 geometry students.

The Treatment

The study was conducted during February and March of 1983 for a period of five weeks. Each class in the study received instruction from the researcher in the normal school setting. A 32K TRS-80 Color Computer and appropriate Logo disk software was utilized to conduct the experiment. The complete microcomputer system was assembled in every class for demonstration and student programming. Instruction was provided on Monday and Wednesday to four classes and on Tuesday and Thursday to the remaining three. Nine hours of programming instruction was provided. Pretest ... posttest administration took ninety minutes. Each student was given a pretest on the first day of the study. The same examination was used as a posttest after completing the five week treatment. If a student was absent for pretesting, he/she was given programming instruction and

allowed to participate in the computer unit, but these scores were not included in the data analysis. Both the pretest and the posttest were scored by the researcher.

The control group was taught programming using the BASIC language for the entire nine hour treatment period. The researcher instructed the students in the syntax of the language, on variables, looping, and arithmetic operations, and in the use of the computer terminal. A list of the objectives for the BASIC programming unit is found in Appendix B. Instruction in BASIC and terminal operation encompassed two and one-half hours of time for the control group.

The experimental group was taught several fundamental commands of the turtle graphics component of Logo including directional commands, repetition commands, and the proper use of variables. The Logo unit was prepared by the writer using information found in "A Beginner's Guide to Logo" (Abelson, 1982) and Mindstorms: Children, Computers, and Powerful Ideas (Papert, 1980). Each class was divided into five subgroups where each subgroup was provided time to explore programming in Logo and to design two Logo programs. The students in the subgroups were allowed to choose the type of program that they wished to create, the only constraint being that the second program had to show repetition and/or use of one or more variables. Instruction and experience in Logo programming composed about 25 percent of

the time devoted to instruction. The students placed the programs that they had designed into the computer's memory, executed the programs, and corrected errors or improved upon the output, whenever needed. (See Appendix F for an example of a typical Logo program and its output.) For the remainder of the treatment, the experimental group had one and one-half hours of instruction in the BASIC language and "hands on" experience in BASIC programming using the same format as the control group.

In order to control access to the computer, each class was divided into five subgroups for the BASIC unit regardless of the size of the class. Each student was randomly assigned by the computer to a subgroup within the class. Group experience is thought to be a valuable means to learn programming because of the practice of one person building upon the ideas of another (Schneider, 1978). The BASIC unit presented to each group was developed by the researcher using Nickles' list of the thirteen essential BASIC commands (Nickles, 1980) and the book BASIC Programming Primer (Waite and Pardee, 1982). Each group received identical instruction in BASIC. Students were encouraged to use all available references. The publications, Pocket Guide to BASIC (Hunt, 1982) and The BASIC Cookbook (Tracton, 1978) were provided for use during the class period. In addition each subgroup was given an identical document that summarized the BASIC commands studied. A copy of the summary of

commands is found in Appendix D.

The source of the problems to program in the BASIC module was the book Problems for Computer Solution by Donald D. Spencer (1977). This book was chosen because it was written specifically to provide programming problems for introductory classes. The pool of problems to be solved by BASIC programming was the same for both the experimental and the control groups.

The problems were typed on cards, and a member of each subgroup selected the card which contained the problem to be solved by that subgroup using a computer program. There were 38 problems in two categories. Problems on green cards required programs that concentrated on input/output statements and that contained no more than one decision component, while the programs designed to solve programs on the yellow cards utilized several programming concepts. Since the subjects were mathematics students, the problems emphasized mathematical concepts of a general nature. If a student needed help with the mathematics required to solve a problem, this information was provided.

Each subgroup completed three problems from the pool of green cards before they solved any on the yellow cards. They completed as many programs as they could in the time allowed. Since access to the computer was limited in that there was only a single machine in each classroom, students were encouraged to observe other subgroups working at the

computer. Thus, students were able to see the computer solutions to problems other than the ones assigned to them. No problems were repeated within the same class.

The Instrument

The evaluation instrument was prepared by the writer after consulting with several teachers of computer programming and with books designed to teach introductory BASIC programming (e.g., Albrecht, Finkel, and Brown, 1978; Blechman, 1981; Faulk, 1982; Fox and Fox, 1983; Golden, 1981; Jacobs, 1983; Nelson, 1981; Sage, 1969; Shelly and Cashman, 1982; Smith, 1970; Waite and Pardee, 1982). This provided a background from which could be prepared a set of objectives for the unit on BASIC programming.

The objectives for the BASIC programming module encompassed the areas of programming style and requirements for correct coding, development of instructions and data structures to perform specified tasks, algorithm development, debugging, and several of the commands directed to the operating system of the computer. Numeric and string variables and one dimensional arrays were the data structures utilized in the unit. The instructions covered in this BASIC module included commands for input and output, arithmetic operations, assignment of values, repetition of code or looping, selection of one of several alternatives, and modularization of code using subroutines. A copy of the

objectives for the BASIC computer programming unit is found in Appendix B.

The researcher perpared a test from these objectives and submitted it to a panel of experts for validation. The panel consisted of Willis J. Bourque, professor of mathematics at the University of Southwestern Louisiana, and Merril T. Mims and Harriet G. Taylor of the Computer Science Department of Louisiana State University.

The instrument was field tested at a high school in Baton Rouge, Louisiana. Item analysis performed on the responses to the examination produced a difficulty index for each item in the range of 0.029 to 0.971 while the index of discrimination was found to be between 0.006 and 0.571. These figures are generally within an acceptable range although several items may be described as being weak. Appendix G is composed of the results of the item analysis on the evaluation instrument. The researcher set the maximum amount of time required by any student to complete the test in the field study as the time to allow for the examination in the actual experiment. Subjects in the study were allowed 45 minutes to complete the pretest and the posttest.

Since students in computer science can be tested on "knowledge of syntax, their ability to read and comprehend a program, or their ability to write logically correct programs" (Lemos, 1979: 53), the researcher included items

that evaluated each of these components. A copy of the examination is included in Appendix A.

The Statistical Procedure

Since the sample was composed of intact mathematics classes, analysis of covariance where the pretest was used as the covariate was the statistical method chosen. Using this procedure results in a statistical correction on post-test scores for factors that may have been present prior to the treatment (Garrett, 1967; Lemos, 1981; Popham, 1967).

The analysis was performed on test data in order to examine the following three hypotheses:

1. There is no significant difference in achievement in BASIC computer programming between secondary mathematics students who are taught programming using BASIC after they learned Logo and those who are taught programming using only BASIC.

2. There is no significant difference in computer programming achievement between secondary geometry students and secondary algebra II students.

3. There is no significant interaction between type of mathematics course (algebra vs. geometry) and type of treatment (BASIC with Logo vs. BASIC alone).

CHAPTER FOUR

Presentation and Analysis of Data

The purpose of this chapter is to indicate the results of statistical analysis on the data obtained from the administration of the BASIC programming test. This examination was given to all participants in the study as a pretest in order to measure the achievement level of the student prior to treatment. These pretest scores were used as the covariate in the subsequent analysis of covariance. After finishing the unit on BASIC programming, each subject completed the same instrument as a posttest. Five weeks of time elapsed between pretest and posttest. The scores of only those students who took both tests were included in the data analysis.

Since achievement in later programming courses is enhanced significantly by a student's having computer experiences in high school (Konvalina, Stephens, and Wileman, 1983), the researcher identified several variables to investigate in order to determine whether they affect achievement in programming at the secondary level. Thus, the statistical analysis was performed to consider the following factors as independent variables:

1. treatment group - instruction in BASIC only,

instruction in BASIC following Logo

2. mathematics class of the student - algebra II, geometry
3. sex of the student - male, female

The score on the BASIC programming posttest was the dependent variable.

The researcher chose to consider the gender of the student as an additional factor above those stated in the hypotheses because research studies performed by Mazlack (1975; 1980) that investigated factors related to aptitude and achievement in introductory computer science included gender as one of the factors to study. Although neither of these research studies measured achievement in BASIC programming, the results indicated that gender was not a significant factor when introductory computer programming was taught at the college level with FORTRAN. In contrast to Mazlack's studies, the researcher sought to investigate the significance of the sex of the student on achievement in BASIC programming at the secondary level.

Results

The statistical technique used to evaluate the results of the experiment was analysis of covariance. A three factor factorial design where the factors considered were group, class, and sex was employed to determine if significant main effects and/or interactions occurred between any

of these factors.

The covariable, pretest, yields an F-value of 9.85 which is significant at the 0.01 level of confidence. Therefore, analysis of covariance using the pretest score as covariate is the proper statistical method to use with the data obtained from this experiment.

The only significant difference found in the adjusted posttest scores was on data from the total sample that was analyzed according to the type of treatment that was employed. Analysis of data from the experimental and the control groups produced an F-value of 4.80 which is significant at the 0.05 confidence level. The adjusted means are 23.528 for the control group (N = 62) that received instruction in BASIC only and 21.435 for the experimental group (N = 69) that was taught Logo before BASIC. Thus, the group that received instruction in BASIC exclusively showed significantly greater achievement on the BASIC programming test than the students in the group that was instructed in Logo before they were taught BASIC.

No significant difference was found between the current mathematics class of the student and his/her achievement in BASIC programming as measured in this study. Analysis of data that related the mathematics class of the student (geometry or algebra II) to posttest scores yielded an F-value of 0.87. Adjusted means are 22.930 for the algebra students (N = 62) and 22.032 for the geometry students

Table 1

Analysis of Covariance on the BASIC Programming Scores of the Sample

Analysis of Covariance

Source of Variation	df	Sum of Squares	F-Value
Group	1	136.098	4.80 *
Class	1	24.684	0.87
Sex	1	8.684	0.31
Group*Class	1	0.025	0.00
Group*Sex	1	26.189	0.92
Group*Class*Sex	1	22.430	0.79
Pretest	1	279.136	9.85 **
Error	122	3457.259	

* significant at 0.05 level

** significant at 0.01 level

(N = 69).

None of the interactions yielded significant results. Table 1 contains the values found by analysis of covariance on posttest scores with the pretest score as the covariate.

An Additional Observation

Since much current literature focuses on sex differences in mathematics and since there is a close relationship between mathematics and computer science via problem solving, the investigator chose to consider the gender of the student as an independent variable in the study. Analysis of covariance using sex as a factor yielded no significant difference in BASIC programming achievement between the males and the females in the sample. The F-value was 0.31. The adjusted posttest mean score for the males (N = 64) was 22.746 whereas that for the females who participated in the study (N = 67) was 22.217.

CHAPTER FIVE

Conclusion and Summary

This study investigated two methods of teaching introductory computer programming to secondary mathematics students. The methods utilized to teach BASIC programming were instructing the students in BASIC programming after they had learned to program the computer using Logo and teaching the BASIC language exclusively. Statistical analysis was performed on the total sample and on subsamples of algebra II and geometry students and of males and females.

Conclusion

The results of the experiment indicate that some factor is responsible for the higher achievement in BASIC programming exhibited by students in the total sample who received instruction in the traditional manner of being taught only BASIC compared to those students instructed in Logo before being taught BASIC, but the current mathematics class or the sex of the student is not the significant component. The researcher speculated that the geometric orientation of students enrolled in that class would cause Logo to be a more effective teaching mode for them than for the algebra students, but the results indicated no significant interaction between programming groups and classes.

Even though the unit consisted of mathematics problems that were solved by computer programming, two traditionally male-dominated areas, no significant difference was found in the BASIC programming achievement of male and of female participants. Future research could center upon the question of achievement of female students related to the sex of their instructor. Results such as the ones found in this study could help to eliminate sex stereotypes that exist in the fields of mathematics and computer science.

Additional research might isolate the causative factor for the difference in BASIC programming achievement exhibited by the total sample. Possible factors to be considered include the age of the student, his/her grade point average, his/her mental ability, and the amount of time devoted to teaching BASIC programming.

Educators are just now realizing the potential of the computer as an instructional tool. Computer education at all levels is in a state of flux because goals and objectives and methods of utilizing the electronic device in the classroom are not well-defined. Much of the problem solving instruction with the computer is found at the college level where this versatile, new machine is even being used for teaching number theory and classical mechanics, two of the oldest disciplines currently taught. There are few reports of similar use of the computer as a problem solving tool in secondary schools. A probable reason for the

lack of computer use at the high school level is the dearth of research on effective methods of teaching programming to secondary students. This study adds some empirical data intended to address this problem.

Suggestions for future research on teaching introductory computer programming using Logo followed by BASIC or employing BASIC alone include experiments that test the affective results of the two teaching methods, the effects of varying the amount of time devoted to Logo programming and altering the time devoted to the entire unit, and the effects of varying the population from which the sample is selected. Groups of adults, of junior high or elementary students, or of high school students from other levels could compose the population from which samples are chosen for additional research.

This experimental design could be used to test for achievement in mathematics when students are taught programming by the two approaches of concern in this study. More formal investigation of the effects of gender as well as the number of BASIC programs completed and the grade level of the student could be conducted with a well-defined population and all samples of large size to allow for a more powerful statistical decision. Much more research needs to be performed in this important field of study.

Summary

The research problem was to investigate the effect of the instructional method employed upon the BASIC computer programming achievement of secondary mathematics students. One group was instructed in BASIC exclusively, and the other group was taught BASIC after they had received instruction in Logo. The experimental-control group, pretest-posttest design was used to determine if there was a significant difference between the two methods used to teach introductory computer programming. Using analysis of covariance, the following hypotheses were tested:

1. There is no significant difference in achievement in BASIC computer programming between secondary mathematics students who are taught programming using BASIC after they learned Logo and those who are taught programming using only BASIC.

2. There is no significant difference in computer programming achievement between secondary geometry students and secondary algebra II students.

3. There is no significant interaction between type of mathematics course (algebra vs. geometry) and type of treatment (BASIC with Logo vs. BASIC alone).

In addition the researcher investigated the relationship between the sex of the student and his/her achievement in BASIC programming.

The subjects of this study were students enrolled in

four geometry classes and three algebra II classes in three St. Mary Parish public schools during the 1983 spring semester. The students in the experimental group were taught computer programming with Logo initially followed by instruction in BASIC programming. The control group learned BASIC programming only. The treatment period was five weeks.

The criterion for achievement in BASIC programming was the score attained on a BASIC programming examination prepared by the experimenter. This instrument served as both pretest and posttest. Validity of the evaluation instrument was established by a panel of experts.

Analysis of covariance was used to test for significant differences between mean scores of the experimental and the control groups on the posttest with the pretest score used as a covariate. This statistical procedure was also used on subsamples composed of geometry students and of algebra students and on ones composed of males and of females. The analysis of subsample data was used to determine if the two methods of teaching BASIC programming would result in significantly different degrees of success depending upon the level of the student's scheduled mathematics class or upon the gender of the student.

Statistical analysis revealed that the difference between adjusted mean scores on the posttest was significant at the 0.05 level with the group that was taught

BASIC exclusively attaining a higher score than the group that was taught Logo and BASIC. There was no significant difference between adjusted posttest mean scores for the sample consisting of students classified according to whether they were enrolled in an algebra II class or in a geometry class. Likewise, no significant difference was found in the adjusted mean scores for the males and the females in the sample.

BIBLIOGRAPHY

Abbott, R. J. The place of computer science in general education. In Proceedings of the 1979 Western Educational Computing Conference. North Hollywood: California Educational Computing Consortium, 1979.

Abelson, H. A beginner's guide to Logo. BYTE, 1982, 7(8), 88+.

Abernathy, K., Piegari, G., and Thorsen, A. L. Computer applications in a finite mathematics course. In D. Harris and B. Collison (eds.), Proceedings of NECC/2 - National Educational Computing Conference 1980. Iowa City: The University of Iowa, 1980.

Albrecht, B., Finkel, L., and Brown, J. R. BASIC for Home Computers: A Self-Teaching Guide. New York: John Wiley, 1978.

Andersen, B. S. Pascal vs. BASIC. In P. Barrette (ed), Microcomputers in K-12 Education. Proceedings of the First Annual K-12 Microcomputers in Education Conference, Southern Illinois University, March 1981. Rockville, Maryland: Computer Science Press, 1982.

Anderson, S. S. Computer-assisted creativity in number theory. Mathematics and Computer Education, 1982, 16, 91-94.

Atchison, W. F. The computer as the object of instruction. In D. C. Johnson and J. D. Tinsley (eds.), Informatics and Mathematics in Secondary Schools. Proceedings of the IFIP TC-3 Working Conference on Informatics and Mathematics in Secondary Schools, Impacts and Relationships, September 1977. Amsterdam: North-Holland Publishing, 1978.

Banagasser, V. Computer literacy for seventh grade. The Computing Teacher, 1983, 10(7), 66-68.

Blechman, F. Programs for Beginners on the TRS-80. Rochelle Park, New Jersey: Hayden Book, 1981.

Bratko, I., Rajkovic, V., and Roblek, B. What should secondary school students know about computers: Analysis of an experiment. In O. Lecarme and R. Lewis (eds.), Computers in Education. Proceedings of the IFIP Second World Conference on Computer Education, 1975. Amsterdam: North-Holland Publishing, 1975.

Carpenter, T. P., Corbitt, M. K., Kepner, H. S., Lindquist, M. M., and Reys, R. E. The current status of computer literacy: NAEP results for secondary students. Mathematics Teacher, 1980, 73, 669-673.

Carpenter, T. P., Corbitt, M. K., Kepner, H. S., Lindquist, M. M., and Reys, R. E. Results from the Second Mathematics Assessment of the National Assessment of Educational Progress. Reston, Virginia: National Council of Teachers of Mathematics, 1981.

Carter, R. The complete guide to Logo. Classroom Computer News, 1983, 3(5), 35-39.

Davis, A. D. Classical mechanics with computer assistance. In D. Harris and B. Collison (eds.), Proceedings of NECC/2 - National Educational Computing Conference 1980. Iowa City: The University of Iowa, 1980.

Dolan, D. T. Montana office of public instruction surveys: Computer education activity in schools. The Computing Teacher, 1982, 9(9), 58.

du Boulay, B., and Howe, J. Re-learning mathematics through Logo: Helping student teachers who don't understand mathematics. In J. Howe and P. Ross (eds.), Microcomputers in Secondary Education: Issues and Techniques. New York: Nichols Publishing, 1981.

EL News. EL survey shows: Computers used more widely in instruction than administration. Electronic Learning, 1982, 1(8), 12.

Faulk, E. How to Write a TRS-80 Program. Chatsworth, California: Datamost, 1982.

Fisher, G. Developing a district-wide computer use plan. The Computing Teacher, 1983, 10(5), 52-59.

Fox, A., and Fox, D. Armchair BASIC: An Absolute Beginner's Guide to Programming in BASIC. Berkeley, California: Osborne/McGraw-Hill, 1983.

Garrett, H. E. Statistics in Psychology and Education. New York: David McKay, 1967.

Gilberts, R. D. The Computer: Extension of the Human Mind. Eugene, Oregon: University of Oregon, 1982. (ERIC Document Reproduction Service No. ED 219 859)

Golden, N. Computer Programming in the BASIC Language. (2nd ed.). New York: Harcourt Brace Jovanovich, 1981.

Gottfried, B. S. Schaum's Outline of Theory and Problems of Programming with BASIC. (2nd ed.). New York: McGraw-Hill, 1982.

Gress, E. K. The future of computer education: Invincible innovation or transitory transformation? The Computing Teacher, 1982, 9(1), 39-42.

Haney, M. R. Review of Karel the Robot - A Gentle Introduction to the Art of Programming, by R. Pattis. The Computing Teacher, 1982, 9(3), 42.

Hansen, T. P., Klassen, D. L., Anderson, R. E., and Johnson, D. C. What teachers think every high school graduate should know about computers. School Science and Mathematics, 1981, 8, 467-472.

Hart, M. Using computers to understand mathematics, four years on. Mathematics Teaching, 1982, 98, 52-54.

Hopper, J. A. A byte of BASIC. In D. Harris and B. Collison (eds.), Proceedings of NECC/2 - National Educational Computing Conference 1980. Iowa City: The University of Iowa, 1980.

Horn, C. E., and Poirot, J. L. Computer Literacy: Problem-Solving with Computers. Austin, Texas: Sterling Swift, 1981.

Howe, J., and Ross, P. Moving LOGO into a mathematics classroom. In J. Howe and P. Ross (eds.), Microcomputers in Secondary Education: Issues and Techniques. New York: Nichols Publishing, 1981.

Hughes, B. Thinking Through Problems. Palo Alto, California: Creative Publications, 1976.

Hunt, R. Pocket Guide to BASIC. Reading, Massachusetts: Addison-Wesley, 1982.

Hunter, B. Computer literacy: 1949-1979. In R. J. Seidel, R. E. Anderson, and B. Hunter (eds.), Computer Literacy: Issues and Directions for 1985. New York: Academic Press, 1982.

Hunter, B., Kastner, C. S., Rubin, M. L., and Seidel, R. J. Learning Alternatives in U. S. Education: Where Student and Computer Meet. Englewood Cliffs, New Jersey: Educational Technology Publications, 1975.

Jacobs, Z. P., French, F. G., Moulds, W. J., Schuchman, J. G. Computer Programming in the BASIC Language. Boston: Allyn and Bacon, 1983.

Jacquet, E. M. Personal communication, March 18, 1983.

Joseph, H. Lesson Plan for a Computer Literacy Unit (3 Weeks). 1979. (ERIC Document Reproduction Service No. ED 216 898)

Knuth, D. E. Computer science and its relation to mathematics. The American Mathematical Monthly, 1974, 81, 323-343.

Konvalina, J., Stephens, L., and Wileman, S. Identifying factors influencing computer science aptitude and achievement. AEDS Journal, 1983, 16, 106-112.

Krulik, S. (ed.). Problem Solving in School Mathematics. Yearbook of the National Council of Teachers of Mathematics. Reston, Virginia: National Council of Teachers of Mathematics, 1980.

La France, J. E. Shall we teach structured programming to children? In D. Harris and B. Collison (eds), Proceedings of NECC/2 - National Educational Computing Conference 1980. Iowa City: The University of Iowa, 1980.

Lemos, R. S. On the measurement of programming language learning. In Proceedings of the 1979 Western Educational Computing Conference. North Hollywood: California Educational Computing Consortium, 1979.

Lemos, R. S. A comparison of non-business and business student test scores in BASIC. SIGCSE Bulletin, 1981, 13, 86-90.

Lias, E. Micro-computers in education. Louisiana Association of School Executives Conference, New Orleans, Louisiana, November 20, 1982.

Lough, T. Logo: Discovery learning with the classroom's newest pet. Electronic Learning, 1983, 2(6), 49-53.

Luehrmann, A. Pre- and post-college computer education. In R. P. Taylor (ed.), The Computer in the School: Tutor, Tool, Tutee. New York: Teachers College Press, 1980a.

Luehrmann, A. Prepared statement on research, development, and planning for computers and the learning society. In R. P. Taylor (ed.), The Computer in the School: Tutor, Tool, Tutee. New York: Teachers College Press, 1980b.

Luehrmann, A. Should the computer teach the student, or vice-versa? In R. P. Taylor (ed.), The Computer in the School: Tutor, Tool, Tutee. New York: Teachers College Press, 1980c.

Luehrmann, A. Technology and science education. In R. P. Taylor (ed.), The Computer in the School: Tutor, Tool, Tutee. New York: Teachers College Press, 1980d.

Luehrmann, A. Computer literacy - What should it be? Mathematics Teacher, 1981, 74, 682-686.

Masat, F. E. An immersion course in BASIC. Journal of Educational Technology Systems, 1981-82, 10, 321-329.

Mavaddat, F. Another experiment with teaching of programming languages. SIGCSE Bulletin, 1981, 13, 40-56.

Mazlack, L. J. Compatability of students from different disciplines and semesters. In O. Lecarme and R. Lewis (eds.), Computers and Education. Proceedings of the IFIP Second World Conference on Computer Education. Amsterdam: North-Holland Publishing, 1975.

Mazlack, L. J. Identifying potential to acquire programming skill. Communications of the ACM, 1980, 23, 14-17.

Morning Advocate. Baton Rouge, Louisiana. April 27, 1983, p. 1, cols. 2-6.

Moursund, D. ACM elementary and secondary schools subcommittee progress report. In D. Harris and B. Collison (eds.), Proceedings of NECC/2 - National Educational Computing Conference 1980. Iowa City: The University of Iowa, 1980.

Muller, L., and Muller, J. Watch out for the turtles! Educational Computer Magazine, 1982, 2(3), 14+.

National Council of Teachers of Mathematics. Position statement on basic skills. Mathematics Teacher, 1978, 71, 147-152.

National Council of Teachers of Mathematics. An Agenda for Action: Recommendations for School Mathematics for the 1980s. Reston, Virginia: National Council of Teachers of Mathematics, 1980.

Nelson, D. E., Burras, D. V., Gillis, E. J., and King R. L. BASIC: A Simplified Structured Approach. Reston, Virginia: Reston Publishing, 1981.

Norris, D. O. Let's put computers into the mathematics curriculum. Mathematics Teacher, 1981, 74, 24-26.

Newell, A., and Simon, H. A. Human Problem Solving. Englewood Cliffs, New Jersey: Prentice-Hall, 1972.

Nickels, H. Teaching teachers the three R's (REM, READ, and RETURN): A BASIC course in instructional computing. In Proceedings of the 1980 Western Educational Computing Conference. North Hollywood: California Educational Computing Consortium, 1980.

Papert, S. Mindstorms: Children, Computers, and Powerful Ideas. New York: Basic Books, 1980.

Papert, S., Watt, D., di Sessa, A., and Weir, S. The Brookline Project Final Report Part II: Project Summary and Data Analysis. Cambridge, Massachusetts: Massachusetts Institute of Technology, Artificial Intelligence Laboratory, 1979. (ERIC Document Reproduction Service No. ED 196 423)

Pattis, R. E. Karel the Robot: A Gentle Introduction to the Art of Programming. New York: John Wiley, 1981.

Patton, R., Ortho, R., and Hopfensperger, P. Computer Literacy for All High School Students. 1981. (ERIC Document Reproduction Service No. ED 216 679)

Pereira-Mendoza, L. Heuristic Strategies utilized by high school students. The Alberta Journal of Educational Research, 1979, 25, 213-220.

Piele, D. T. Computer assisted problem solving in mathematics. In The Computer: Extension of the Human Mind. Eugene, Oregon: University of Oregon, 1982. (ERIC Document Reproduction Service No. ED 219 859)

Polya, G. How to Solve It. (2nd ed.). Princeton, New Jersey: Princeton University Press, 1957.

Polya, G. Mathematical Discovery (Vol. 1). New York: John Wiley, 1962.

Polya, G. Mathematical Discovery (Vol. 2). New York: John Wiley, 1965.

Popham, W. J. Educational Statistics: Use and Interpretation. New York: Harper and Row, 1967.

Reetz, G. Why Johnny can't compute. Electronic Learning, 1983, 2(7), 24+.

Robison, F. G., Tuckle, J., and Brison, D. W. Inquiry Training: Fusing Theory and Practice. Profiles in Practical Education, The Ontario Institute for Studies in Education. Toronto: The Ontario Institute for Studies in Education, 1972.

Rubinstein, M. F. Explicit heuristic training as a variable in problem-solving performance. Journal for Research in Mathematics Education, 1975, 10, 173-187.

Sage, E. R. Problem-Solving with the Computer. Newburyport, Massachusetts: Entelek, 1969.

Schneider, G. M. The introductory programming course in computer science--ten principles. SIGCSE Bulletin, 1978, 10, 107-114.

Shane, H. G. The silicon age and education. Phi Delta Kappan, 1982, 63, 303-308.

Shelly, G. B., and Cashman, T. J. Introduction to BASIC Programming. Brea, California: Anaheim Publishing, 1982.

Smith, R. E. Discovering BASIC: A Problem Solving Approach. Rochelle Park, New Jersey: Hayden Book, 1970.

Sobol, T., and Taylor, R. The Scarsdale project: Integrating computing into the K-12 curriculum. In D. Harris and B. Collison (eds.), Proceedings of NECC/2 - National Educational Computing Conference 1980. Iowa City: The University of Iowa.

Solomon, C. J. Leading a child to a computer culture. In The Papers of the ACM SIGCSE-SIGCUE Technical Symposium on Computer Science and Education 1975. New York: Association for Computing Machinery, 1975.

Soloway, E., Lochhead, J., and Clement, J. Does computer programming enhance problem solving sbility? Some positive evidence on algebra word problems. In R. J. Seidel, R. E. Anderson, and B. Hunter (eds.), Computer Literacy: Issues and Directions for 1985. New York: Academic Press, 1982.

Spencer, D. D. Problems for Computer Solution. Ormond Beach, Florida: Camelot Publishing, 1972.

Taylor, H., and Poirot, J. L. State certification of computer science teachers progressing slowly. Electronic Learning, 1983, 2(7), 16-18.

Tracton, K. The BASIC Cookbook. Blue Ridge Summit, Pennsylvania: Tab Books, 1978.

Waite, M., and Pardee, M. BASIC Programming Primer. (2nd ed.). Indianapolis: Howard W. Sams, 1982.

Watt, D. The brouhaha over computers in the classroom. Popular Computing, 1982a, 1(7), 36-45+.

Watt, D. Logo in the schools. BYTE, 1982b, 7(8), 116-120.

Watt, D. Should children be computer programmers? Popular Computing, 1982c, 1(11), 130-133.

Wicklegren, W. A. How to Solve Problems. San Francisco: W. H. Freeman, 1974.

Winogard, T. Beyond programming languages. Communications of the ACM, 1979, 22, 391-401.

Wold, A. L. What is a programming language? Classroom Computer News, 1983, 3(5), 46-49.

APPENDICES

APPENDIX A

Instrument used as Pretest/Posttest on BASIC Computer Programming

Part 1 **MULTIPLE CHOICE** Darken the space on the answer sheet that contains the letter of the correct response.

1. The BASIC statement that is never executed by the computer but that is used by the programmer to make the program easier to read is

   (A) PRINT (C) READ (E) none of these
   (B) COMMENT (D) REM

2. In BASIC which of the following symbols is used for multiplication?

   (A) X (C) ** (E) all of these
   (B) * (D) •

3. Every line of a BASIC program must contain

   (A) an equal sign (C) a statement number (E) none of these
   (B) the word LET (D) an equation

4. If you wanted to give N a value of 100, which type of statement would you use?

   (A) SET (C) PRINT (E) none of these
   (B) LET (D) LIST

5. In order to execute a program that is stored in the memory of the computer, which command must be typed and entered?

   (A) LOAD (C) RUN (E) none of these
   (B) SAVE (D) EXECUTE

6. Locating and correcting errors in a computer program is called

   (A) structuring (C) documenting (E) debugging
   (B) reverifying (D) tidying

7. The command used to retrieve a program stored on a diskette is

   (A) READ (C) LOAD (E) none of these
   (B) PRINT (D) SAVE

8. To instruct the computer to accept alphabetic information such as a name from the keyboard, which of the following statements would be used?

   (A) 30 INPUT A (C) 30 INPUT A$ (E) 30 READ A
   (B) 30 INPUT "A" (D) 30 READ A$

MULTIPLE CHOICE --2--

9. If the statement LET N$ = "3 + 4" is followed by the statement PRINT N$ what would be the output?

(A) 3 + 4 (C) "3 + 4" (E) none of these
(B) 7 (D) "7"

10. Which statement group is associated with the input of data?

(A) READ/DATA (C) INPUT/DATA (E) all of these
(B) GET/DATA (D) A and C only

11. What is the output of the following program?

```
10  LET A = 5
20  LET B = 6
30  LET C = 7
40  PRINT B  ;  A
50  END
```

(A) 5 6 7 (C) 5 6 (E) 6 5
(B) 5 6 (D) 6 5

12. Every program must contain a statement called

(A) END (C) LET (E) WRITE
(B) BEGIN (D) FOR

13. Consider the following programming segment:

```
100  PRINT "WHAT IS YOUR NAME";
110  INPUT N$
```

What will be the value of N$ when the program is executed?

(A) whatever is entered from the keyboard (C) "WHAT IS YOUR NAME" (E) N
(B) nothing (D) zero

14. In order to print the values of six variables in three columns on each of two lines of output placed directly under each other, one could use which of the following sequence of statements?

(A)
```
10  PRINT A,B,C
20  PRINT X,Y,Z
```

(B)
```
10  PRINT A;B;C;
20  PRINT X;Y;Z
```

(C)
```
10  PRINT A,B,C;
20  PRINT X,Y,Z
```

(D)
```
10  PRINT A,B;C
20  PRINT X;Y,Z
```

(E) all of these

MULTIPLE CHOICE --3--

15. In order to print the word "HERE" in the twentieth column from the left side of the screen, one would write

(A) 50 PRINT(20);"HERE"
(B) 50 PRINT"HERE";(20)
(C) 50 PRINT TAB(20);"HERE"
(D) 50 PRINT"HERE";TAB(20)
(E) 20 PRINT "HERE"

16. Which of these pre-programmed functions is available in BASIC?

(A) ABS
(B) SQR
(C) INT
(D) RND
(E) all of these

17. What is the output of the following program?

```
10  READ A
15  DATA -1,3,6,5
20  IF A > 5 THEN GOTO 50
30  PRINT "ABC,";
40  GOTO 10
50  PRINT "XYZ,";
60  END
```

(A) ABC,XYZ,XYZ,XYZ
(B) XYZ,ABC,XYZ,ABC
(C) XYZ,XYZ,ABC,
(D) ABC,ABC,XYZ,
(E) ABC,ABC,XYZ,ABC

18. Consider the following program:

```
10  FOR I = 1 TO 5
20  PRINT "GO TEAM"
30  NEXT I
40  PRINT "RAH! RAH! RAH!"
50  END
```

How many lines of output will be printed?

(A) 2
(B) 5
(C) 6
(D) 8
(E) 13

19. In order to display the program that is currently in the computer's memory, the command to be used is

(A) DISPLAY
(B) SHOW
(C) SAVE
(D) PRINT
(E) LIST

MULTIPLE CHOICE --4--

20. In the following problem which operation will be done first in BASIC?

13 + 2 - 6 * 5 / 10 + 3

(A) 13 + 2
(B) 2 - 6
(C) 6 * 5
(D) 5 / 10
(E) 10 + 3

Part 2 **SHORT ANSWERS** Indicate your response on the answer sheet in the proper section of part 2.

1. Write a BASIC statement to perform the following calculation:

$$A = \frac{x + y + z}{6}$$

2. Indicate if each of the following variable names are valid or not valid in BASIC by choosing the appropriate term on the answer sheet.

(A) A
(B) AB
(C) A$
(D) 2B
(E) B2
(F) B
(G) 12A

3. Indicate if each of the following BASIC statements or program segments is correct or if it contains an error by choosing the appropriate response on the answer sheet.

```
(A) 10  PRINTT A
(B) 10  READ, N
(C) 10  LET X + Y = A
(D) 10  INPUT X$
(E) 10  IF X<>Y THEN PRINT "GOOD"
(F) 10  LET A = LW
(G) 10  FOR J = 5 TO 23 STEP 4
(H) 10  DIM A(100)
(I) 10  IF X<Y OR Y>2 THEN GOTO 50
(J) 10  REM PROGRAM TO ADD NUMBERS
    20  PRINT "ENTER TWO NUMBERS TO ADD TOGETHER"
    30  INPUT N1, N2
    40  S = N1 - N2
    50  PRINT S
    60  END
(K)  5  LET A = 5
    10  LET Y = A - A
    20  LET Z = 2 + A
    30  LET X = Z/Y
    40  END
```

Part 3 **PROGRAM ANALYSIS** Indicate your response on the answer sheet in the section labeled "Part 3."

Indicate the output of each of the following programs.

1.
```
10  LET C = 1
20  LET S = 0
30  IF C > 5 THEN 80
40    LET S = S + C
50    PRINT S
60    LET C = C + 1
70  GOTO 30
80  END
```

output:

2.
```
10  FOR I = 1 TO 3
20     READ N
30     GOSUB 100
40     PRINT A,  B
50  NEXT I
60  GOTO 140
100 LET A = N * I
110 LET B = N↑2
120 RETURN
130 DATA 5, 2, 6
140 END
```

output:

3.
```
10  DIM A(10)
20  FOR J = 1 TO 10
30    READ A(J)
40  NEXT J
50  LET T = 0
60  FOR J = 2 TO 5
70    T = T + A(J)
80  NEXT J
90  PRINT T
100 DATA 6,-3,2,5,0
110 DATA -2,1,8,-1,4
120 DATA 6,3,9,-4
130 END
```

output:

Part 4 **PROGRAM WRITING** Write your program on the second sheet of the answer section. Be sure to place your student number and class code on that page as well as on the first page of the answer sheet.

Write a program that will accept a student's name and three test scores for each of five students. Determine each student's average and print out each student's name, test scores, and average. A student passes if his average is 70% or greater; otherwise he fails. On a separate line indicate how many of the students passed and how many failed. Be sure to label all output.

Each test has a possible score of 100 points. Use the following entries as data:

NAME	TEST 1	TEST 2	TEST 3
Abbot Joe	89	100	97
Adams Jane	93	95	92
Brown Kate	63	65	71
Caldwell Jeff	83	75	79
Counts Kevin	52	59	80

ANSWER SHEET

Part 1 - Multiple Choice

1.	(A)	(B)	(C)	(D)	(E)	11.	(A)	(B)	(C)	(D)	(E)
2.	(A)	(B)	(C)	(D)	(E)	12.	(A)	(B)	(C)	(D)	(E)
3.	(A)	(B)	(C)	(D)	(E)	13.	(A)	(B)	(C)	(D)	(E)
4.	(A)	(B)	(C)	(D)	(E)	14.	(A)	(B)	(C)	(D)	(E)
5.	(A)	(B)	(C)	(D)	(E)	15.	(A)	(B)	(C)	(D)	(E)
6.	(A)	(B)	(C)	(D)	(E)	16.	(A)	(B)	(C)	(D)	(E)
7.	(A)	(B)	(C)	(D)	(E)	17.	(A)	(B)	(C)	(D)	(E)
8.	(A)	(B)	(C)	(D)	(E)	18.	(A)	(B)	(C)	(D)	(E)
9.	(A)	(B)	(C)	(D)	(E)	19.	(A)	(B)	(C)	(D)	(E)
10.	(A)	(B)	(C)	(D)	(E)	20.	(A)	(B)	(C)	(D)	(E)

- -

Part 2 - Short Answers

1. _____

2.

	valid	not valid
(A)	( )	( )
(B)	( )	( )
(C)	( )	( )
(D)	( )	( )
(E)	( )	( )
(F)	( )	( )
(G)	( )	( )

3.

	correct	error
(A)	( )	( )
(B)	( )	( )
(C)	( )	( )
(D)	( )	( )
(E)	( )	( )
(F)	( )	( )
(G)	( )	( )
(H)	( )	( )
(I)	( )	( )
(J)	( )	( )
(K)	( )	( )

- -

Part 3 - Program Analysis

1. output: _____

2. output: _____

3. output: _____

- -

STUDENT NUMBER _____

CLASS CODE _____

ANSWER SHEET

Part 4 PROGRAM WRITING

- -

STUDENT NUMBER _____

CLASS CODE _____

- -

APPENDIX B

Objectives for BASIC Programming Unit

OBJECTIVES

After a unit in introductory BASIC programming the student shall recognize, understand, and be able to use correctly each of the following programming concepts:

1) elements of programming style including documentation, indention of code, and use of the REM statement

2) requirements for proper programming coding such as a statement number for each line of code and the presence of an END statement

3) input/output statements using the keywords READ, DATA, INPUT, and PRINT including methods of formatting output employing the semicolon, comma, and TAB

4) assignment statements using LET

5) available arithmetic operations and the hierarchy of implementation of these operations

6) selection statements such as IF THEN and GOTO as well as the role of the relational operators and the Boolean operators

7) subroutines and their functions and the GOSUB and RETURN statements

8) repetition statements such as FOR NEXT or the establishment of a loop using a counter

9) library functions including ABS, SQR, INT, RND, and available trigonometric functions

10) establishing and manipulating one-dimensional arrays using the DIM statement and subscripted variables

11) the meaning of the system commands RUN, LIST, LOAD, and SAVE

12) distinction between numeric and string variables and constants

13) developing a method to represent an algorithm prior to coding

14) debugging a program

APPENDIX C

Letter Granting Permission for Student Participation in the Study

_____
(Student's Name)

Dear Parent,

A graduate student at Louisiana State University, I am doing a study on methods of teaching computer programming. Since computers will be such an important part of the lives of all of us, it is desirable for high school students to have experience using this electronic machine. One way to gain this experience is by learning to program.

Your child's mathematics class has been selected to participate in the research. The assistant superintendent of curriculum and instruction, school principal, and mathematics teacher have agreed to allow me to conduct the research in the school. I need your permission for your child to participate in the study.

Each participant will receive ten hours of instruction in programming a computer at school in his mathematics class. A teacher in St. Mary Parish schools for twenty-one years, I will teach the unit on computer programming. A test will be given before and after the instruction in order to determine how much programming was learned in that period of time. No names or individual scores will be reported nor will any school or class be identified. There will be no homework assigned to any participant, and the student will not receive a grade or credit in computer programming for taking part in this study.

Thank you for your consideration. Please sign this permission slip and have your child return it as soon as possible.

Sincerely,

Diane B Calamari

Diane B. Calamari

_____
(Parent's Name)

APPENDIX D

Summary of BASIC Commands

Summary of BASIC Commands

1) Every line of a BASIC program must have a line number. This number determines the order in which the statements of the program are executed.

2) Every program must end with a statement called END.

3) OUTPUT: command is PRINT. Use quotation marks around a word or phrase if you wish it to be output exactly as you write it. If you PRINT a mathematical expression or a variable without any quotation marks, the value of the expression or the variable will be printed on the screen.

4) MATHEMATICAL OPERATIONS: computer can perform addition, subtraction, multiplication, and division; can also raise a number to a power (use upward arrow key). Multiplication and division are performed from left to right followed by all addition and subtraction from left to right. The order of operations can be changed by using parentheses.

5) When commas are used to separate data, the output is spaced far apart; to have output spaced close together, use semicolons to separate data or variables.

6) VARIABLES: numeric variables refer to numbers (integers or decimal numbers); string variables refer to words or special symbols. Variable names must begin with a letter of the alphabet; then it can have any number or another letter; string variables must end with a $.

7) LET: used to assign values to variables and to assign a value to an expression or formula involving calculations.

8) INPUT: used to enter data from the keyboard. Need to precede an INPUT statement with a PRINT statement to tell the user what is to be input. If more than one variable value is input in the same statement, each value should be separated by commas.

9) TAB: used to space output on the screen. For example, PRINT TAB (15); "Name" places the "N" of Name in the fifteenth column of the screen.

10) REM: stands for remark and is used to tell the reader what the program is about. REM is not executable, but it appears in the listing of the program.

11) RUN, LIST, NEW: these commands are used to control what the computer does at a particular time. RUN means to execute the program that is in the computer's memory. LIST means to display the program as it was typed in at the keyboard. NEW is used to clear the memory of the computer before entering another program.

12) LOOPING: occurs whenever you want to process more than one set of data with the same program. Most loops are established by using FOR...NEXT or IF...THEN.
FOR...NEXT: This loop uses a FOR statement and a NEXT statement as the borders of the loop. If one programs the following:

```
10 FOR I = 1 to 45
20   some
30   program
40   statements
50 NEXT I
```

the loop containing "some program statements" will be run 45 times. If you want the loop executed a special number of times that is not always increased by 1 then use:
10 FOR 1 = 2 TO 36 STEP 2 and cause I to have values of 2, 4, 6,...,36.
IF...THEN: when the "if" part is true, the "then" is executed; otherwise, the statement following the IF...THEN is executed. This statement can be used to compare using =, <, >, <=, >=, or <>. The connectives AND and OR can be used in IF...THEN statements.
GOTO: this statement is used to unconditionally direct execution of the program to another statement identified by its number.

13) The computer has certain built-in functions that can be useful to the programmer. These include ABS, INT, RND, SQR. ABS(X) returns the absolute value of X; INT(Y) drops any decimal part of Y so INT(3.59) = 3, INT(2,2) = 2 or INT(5) = 5 - this is called truncation. RND(Z) produces a random number that is positive and has a value not greater than Z; SQR(A) produces the square root of A if A is not negative (error is A is negative).

14) SUBROUTINES: used whenever a portion of the program must be repeated. The subroutine is called in the main program with a GOSUB statement number command and the subroutine is ended with a RETURN statement.

15) ARRAYS are used to keep several bits of data that refer to the same data set together. Arrays must be dimensioned with a DIM statement before any reference can be made to the array. Elements of an array are denoted by using subscripted variables like A(1), A(2), etc.

16) READ/DATA is used to input data within the program (as opposed to INPUT which is used to receive data from the keyboard). A READ statement must have a DATA statement to which it refers. The computer will READ the data items in the order which they are typed in the program. This is used when the input is not too variable.

note: This is a brief summary of the BASIC commands that we have studied.

APPENDIX E

List of Heuristic Strategies

List of Heuristic Strategies

source: Polya, 1957: xvi-xvii

UNDERSTANDING THE PROBLEM

First. You have to understand the problem.
What is the unknown? What are the data? What is the condition?
Is is possible to satisfy the condition? Is the condition sufficient to determine the unknown? Or is it insufficient? Or redundant? Or contradictory? Draw a figure. Introduce suitable notation. Separate the various parts of the condition. Can you write them down?

DEVISING A PLAN

Second. Find the connection between the data and the unknown. You may be obliged to consider auxillary problems if an immediate connection cannot be found. You should obtain eventually a plan of the solution. Have you seen it before? Or have you seen the same problem in a slightly different form?
Do you know a related problem? Do you know a theorem that could be useful?
Look at the unknown! And try to think of a familiar problem having the same or a similar unknown.
Here is a problem related to yours and solved before.
Could you use it? Could you use its result? Could you use its method? Should you introduce some auxillary element in order to make its use possible.
Could you restate the problem? Could you restate it still differently? Go back to the definitions.
If you cannot solve the proposed problem, try to solve first some related problem. Could you imagine a more accessible related problem? A more general problem? A more special problem? An analogous problem? Could you solve a part of the problem? Keep only a part of the condition, drop the other part; how far is the unknown then determined, how can it vary? Could you derive other data appropriate to determine the unknown? Could you change the unknown or the data, or both if necessary, so that the new unknown and the new data are nearer to each other?
Did you use all the data? Did you use the whole condition? Have you taken into account all essential notions involved in the problem?

CARRYING OUT THE PLAN

Third. Carry out your plan.
Carrying out your plan of the solution, check each step. Can you see clearly that the step is correct? Can you prove that it is correct?

LOOKING BACK

Fourth. Examine the solution obtained. Can you check the result? Can you check the argument?
Can you derive the result differently? Can you see it at a glance?
Can you use the result, or the method, for some other problem?

APPENDIX F

Logo Commands and Sample Logo Programs

Summary of Logo Commands

FORWARD N (or FD N): moves the cursor N units in the direction in which it is pointing leaving a trail on the screen

RIGHT A (or RT A): rotates the cursor A degrees clockwise with no lateral movement

LEFT A (or LT A): rotates the cursor A degrees counter-clockwise with no lateral movement

REPEAT N(statements): executes the program statements within the parentheses N times -- loop structure

HIDE TURTLE (or HT): causes the cursor to disappear from the screen

SHOW TURTLE (or ST): places the cursor at its current position on the screen

The format of a Logo program is as follows:

```
TO program name
   program statement(s)
END
```

```
TO SQUARE
  FORWARD 20
  RIGHT 90
  FORWARD 20
  RIGHT 90
  FORWARD 20
  RIGHT 90
  FORWARD 20
  RIGHT 90
END
```

or

```
TO SQUARE
  FD 20
  RT 90
  FD 20
  RT 90
  FD 20
  RT 90
  FD 20
  RT 90
END
```

or

```
TO SQUARE
  REPEAT 4(FD 20  RT 90)
END
```

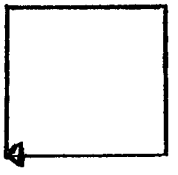

Figure 1 Output of Logo Program Named SQUARE

```
TO DESIGN
  REPEAT 24(SQUARE RT 15)
  HT
END
```

or

```
TO DESIGN   NUM:36, TURN:15
  REPEAT NUM(SQUARE  RT TURN)
  HT
END
```

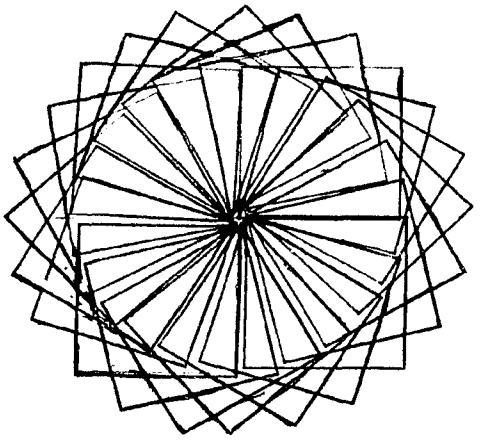

Figure 2 Output of Logo Program Named DESIGN .

```
TO BIGSQ
   REPEAT 4(FD 30  RT 90)
END

TO DESIGN2
  REPEAT 36(BIGSQ  RT 10)
  HT
END
```

or

```
TO DESIGN2 NUM:36, TURN:10
  REPEAT NUM(BIGSQ  RT TURN)
  HT
END
```

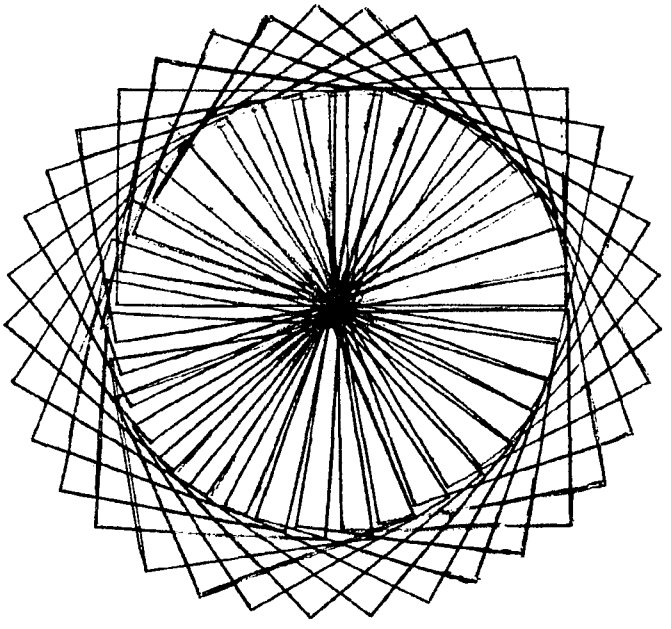

Figure 3 Output of Logo Program Named DESIGN2

APPENDIX G

Item Analysis of the Evaluation Instrument

Part of the Test	Item Number	Difficulty Index	Index of Discrimination
I	1	0.414	0.486
	2	0.900	0.143
	3	0.929	0.143
	4	0.914	0.171
	5	0.457	0.229
	6	0.200	0.400
	7	0.486	0.514
	8	0.457	0.571
	9	0.300	0.314
	10	0.114	0.171
	11	0.286	0.171
	12	0.929	0.143
	13	0.614	0.257
	14	0.488	0.514
	15	0.571	0.571
	16	0.414	0.429
	17	0.343	0.400
	18	0.386	0.600
	19	0.843	0.200
	20	0.643	0.600
II	1	0.100	0.200
	2a	0.900	0.200
	2b	0.657	0.343
	2c	0.886	0.171
	2d	0.529	0.429
	2e	0.486	0.229
	2f	0.800	0.229
	2g	0.557	0.486
	3a	0.971	0.057
	3b	0.600	0.543
	3c	0.300	0.371
	3d	0.829	0.257
	3e	0.529	0.257
	3f	0.429	0.114
	3g	0.314	0.114
	3h	0.657	0.229
	3i	0.729	0.257
	3j	0.457	0.286
	3k	0.429	0.514
III	1	0.029	0.057
	2	0.100	0.200
	3	0.001	0.006
IV	program	0.029	0.057

VITA

Mary Diane Bodin Calamari was born in Franklin, Louisiana. She attended public elementary and secondary schools there and is a graduate of Franklin High School.

In 1957 she entered the University of Southwestern Louisiana as a chemistry major. She has received a Bachelor of Science in chemistry from Louisiana State University in 1961, a Master of Education in 1969 from the University of Southwestern Louisiana, and an Educational Specialist degree from the University of Southwestern Louisiana in 1978.

She has been employed in the public and private secondary schools of St. Mary Parish for twenty-three years as a classroom teacher. During that time she has taught mathematics, science, and computer science.

EXAMINATION AND THESIS REPORT

Candidate: Mary Diane Bodin Calamari

Major Field: EDUCATION

Title of Thesis: A COMPARISON OF TWO METHODS OF TEACHING COMPUTER PROGRAMMING TO SECONDARY MATHEMATICS STUDENTS

Approved:

B. J. Brown
Major Professor and Chairman

William Joyce
Dean of the Graduate School

EXAMINING COMMITTEE:

Steven N Bleau

John A Hildebrand

Donald H Kaff

Richard G. Lomax

Anthony W. Romano

Date of Examination:

July 13, 1983