To the Graduate Council:

I am submitting herewith a dissertation written by Ahmed Sahab Al-Ghamdi entitled "Comparison of Two Teaching Methods for Learning A Computer-Programming Language." I have examined the final copy of this dissertation for form and content and recommended that it be accepted in partial fulfillment of the requirements for the degree of Doctor of Philosophy, with a major in Education.

John I. Matthews, Major Professor

We have read this dissertation
and recommend its acceptance:

Accept for the Council:

Vice Provost
and Dean of The Graduate School

COMPARISON OF TWO TEACHING METHODS FOR LEARNING A COMPUTER-PROGRAMMING LANGUAGE

A Dissertation

Presented for the

Doctor of Philosophy

Degree

The University of Tennessee, Knoxville

Ahmed Sahab Al-Ghamdi

December 1987

ACKNOWLEDGMENTS

The support and assistance of many people have been essential to the completion of this study. It is my pleasure to take this opportunity and formally acknowledge the guidance and direction of my major professor and committee chairman, Dr. John I. Matthews. His continuous support, encouragement, technical expertise, and friendship have had a tremendous influence on me and the development of this research project.

I would like to extend my appreciation and gratitude to my committee members, Dr. Clifton Campbell, Dr. Walter Cameron, and Dr. Gordon Sherman, for their help, hard work, and useful suggestions.

Thanks and appreciation go to the User Services staff at the University of Tennessee Computing Center for their support, encouragement, and friendship. Special thanks goes to Bruce Delaney the manager of User Services.

Much gratitude and very special thanks are owed to all of my family members, especially my parents for their understanding, financial support, emotional support, and love.

ABSTRACT

The purpose of this study was to determine, through experimentation, if a structured, conceptual method of instruction, combined with assigned structured supplementary examples of computer program routines would produce higher test scores and better written programs than the traditional lecture and demonstration methods of instruction.

A quasi-experimental, pretest-posttest control group, design was used in the study. The population consisted of students enrolled in the BASIC computer-programming courses taught in Technological and Adult Education Department (TAE) at the University of Tennessee. The sample for the initial study was comprised of the 22 students attending the two sections of the course TAE 5150 during the Spring quarter, 1987. The study was replicated in the first session of the Summer quarter with a sample of 19 students attended the same course TAE 5150.

Fifteen supplementary aid examples and four instruments were developed for this study; (a) the demographic survey, (b) the knowledge test, (c) the final exam, (d) and the program checklist.

The data were collected and then analyzed using the t-test to determine if there had developed a significant difference in students' achievement and programming skills between the experimental and control groups.

The major findings of this study included:

1. In the initial study, students' pretest-posttest mean-gain scores were significantly higher for the experimental group than that for the control group. However, in the replication study, no apparent significant difference was found between the two groups.

2. In the initial study, the students' ability to read and trace the flow of a program was significantly better for the experimental group than that of the control group. However, the replication study did not produce the same result.

3. At the end of the study and its replication, there was no significant difference in programming skills between the experimental and the control group.

It was concluded, that the lecture and demonstration method of computer-programming language instruction, supplemented with assigned structured aid examples, does not significantly enhance students' achievement and programming skills any more than the traditional lecture and demonstration method alone.

TABLE OF CONTENTS

CHAPTER PAGE

PAGE

LIST OF TABLES

TABLE PAGE

LIST OF FIGURES

CHAPTER I

INTRODUCTION

There has been only a short 12-year history of the use of microcomputers in education since its emergence in 1975. The demand for and growth in the use of microcomputers has been tremendous since 1980. "Though a recent invention, computers are fast becoming a part of everyday life" (Wiggins, 1984, p. 1).

Educators believe that computer programming is an important skills to today's technological society. Furthermore, it has been the hope and expectation of many educators, psychologists, and computer scientists that computer programming can be a powerful mean by which children can develop their problem solving skills (Ehrlich et al., 1984). Thinking and problem solving skills would be reflected in other disciplines such as mathematics and science as well as computer science (Kurland et al., 1984).

Educators have stressed the integration of computer education and computer-programming courses into the school curriculum. It is believed that computer literacy, specially computer programming, gives students a greater chance to find employment in todays schools and computer market (Ehrlich et al., 1984). Educational microcomputer-programming courses at the university, in Colleges of Education, Business, Engineering, and Home Economics has been stressing the needs and demands for computer programming. They are providing courses to teach computer programming and computer-programming applications.

Since research has established that different approaches to teaching differ in effectiveness, there is a need to determine the most effective, reliable, and productive approach to teaching computer-programming language courses. What best motivates students to produce good computer programs that are correct, understandable, and reflect

the structure of the problems the programs represent, is not well documented in the literature.

Statement of the Problem

Lecture and demonstration methods of instruction used in computer-programming courses do not appear to develop adequate conceptual relationships between the instruction and actual programming projects without significant anxiety. The problem may be related to how hands-on interactive experience with the computer in writing programs is coordinated with the lecture and demonstration methods and term-project development. Problem-solving skills and their relation to a creative attitude seem to be poorly developed in most programming instruction. Will a structured approach utilizing these concepts increase the level of understanding to produce more complete and creative programs?

Purpose of the Study

The purpose of this study was to determine, through experimentation, if a structured, conceptual method of instruction, combined with assigned structured supplementary examples of computer-program routines, would produce higher test scores and better written programs than the traditional lecture and demonstration methods of instruction.

The objectives of this study were to:

1. Compare the performance, based on pretest-posttest gain scores, between students who received supplementary BASIC programming exercises, in addition to traditional instruction, and those who received traditional instruction alone.

2. Compare the quality of the programs written by students who received supplementary BASIC programming exercises, in addition to traditional instruction, with those who received traditional instruction alone.

Theoretical Basis of The Study

This study is based on the premise that students learn best when they are actively participating in the learning process. Assigned structured exercises, developed and provided to students, should give them a chance to (a) read computer-language programs, (b) understand good programming techniques, (c) see how computer language syntax and statements fit into a complete computer program, and finally (d) develop a hands-on experience with lower anxiety, high motivation, and better programming skills.

Many instructional theorists suggest that individuals have their own learning style and can learn best when the learning is based on and directly related to the learner's experience and participation, and when there is some choice concerning personal behavior (Walter, 1981, p. 281). Woodworth and Scholosberg (1954), in their studies of experimental psychology, found that students best learn a skill through practicing that skill and seeing the result of their work. Lucas and Thompson (1974), in his comparative study, tended to support the idea that more participation in the learning process by the learner is a feasible and productive alternative to the lecture and discussion instructional technique.

Anderson and Faust (1973, p. 429), using research evidence available in the field of motivation and anxiety, concluded:

> Anxious students (and, presumably, very strongly motivated students, no matter what the exact nature of the motivation) will do best in instructional programs allowing them to master each component skill thoroughly before proceeding and in which standards of performance are made very clear.

McKeachie (1963) said many educators believe anxious students do best in a highly structured learning environment which reduces their anxiety level because they then know what is expected of them.

Students learning computer programming should be exposed to structured practical exercises that serve as concrete models, and reflect techniques of good programming. Students learning a computer-programming language must acquire specific concepts, facts, rules, and skills that help them to understand and conceptualize syntax, statements, and programs. Chand (1978) noted that drill and practice is a well accepted technique for learning new material and is essential in the process of learning the art and craft of computer programming. Walter (1981, p. 184) noted that:

> The activity provides a contest and source of feedback and conditioning. Interpersonal feedback about consequences, when paired with clear criteria, serves a negative feedback function enabling participates to focus lightly on learning objectives. This negative feedback is positively reinforced because knowledge of results leads to improved performance and an intrinsic sense of achievement.

Bohl (1978, p. 124) stated that programmers who study program listings of well-documented code produced by skilled programmers can learn much about coding from them.

Some theoretical notions about learning and the learning process, as cited, and which research seemed to support, provided the theoretical framework of this study. They are as follow:

- Lower anxiety students have a higher performance rate than highly anxious students.
- Problem-solving and computer-programming language skills are best acquired and strengthened through practical experiences.

- "Strong motivation facilitates learning and performance when the desired behavior is already established or can easily be acquired" (Anderson and Faust, 1973, p. 428).
- "Programmers can learn much by studying examples" (Bohl, 1978, p. 123).
- Feedback is an important activity in the learning process.

Null Hypotheses to be Tested

1. There is no significant difference in the performance of students who received Supplementary aid examples, as reflected by the pretest-posttest mean-gain scores, and those who received the traditional lecture and demonstration instruction only.

2. There is no significant difference in the general conceptual understanding of students who received supplementary aid examples, as reflected by the mean-gain scores from part 1 of the pretest-posttest, and those who received the traditional lecture and demonstration instruction only.

3. There is no significant difference in the debugging ability of students who received supplementary aid examples, as reflected by the mean-gain scores from part 2 of the pretest-posttest, and those who received the traditional lecture and demonstration instruction only.

4. There is no significant difference in the ability to read and understand a computer program flow of students who received supplementary aid examples, as reflected by the mean-gain scores from part 3 of the pretest-posttest, and those who received the traditional lecture and demonstration instruction only.

5. There is no significant difference in the quality of programs written by students who received supplementary aid examples, as determined by the final projects mean scores, and those who received the traditional lecture and demonstration instruction only.

6. There is no significant difference in the performance of students who received supplementary aid examples, as reflected by the posttest mean scores, and those who received the traditional lecture and demonstration instruction only.

7. There is no significant difference in the performance of students who received supplementary aid examples, as reflected by the mean scores of the final exam, and those who received the traditional lecture and demonstration instruction only.

Population

The target population of this study consisted of students enrolled in the Department of Technological and Adult Education (TAE) computer-programming language courses. However, the sample for this study was comprised of the students attending the TAE 5150 course during the Spring quarter, 1987 and first session of the Summer quarter, 1987. There were two sections of TAE 5150 taught during the Spring quarter, 1987 and two sections of the same course taught during first session of the Summer quarter, 1987. The two sections taught during the first session of the Summer quarter, 1987, served as the sample for the replication study. The number of students enrolled in each course section is shown in Table 1. All sections of TAE 5150 were taught by the same instructor.

> The TAE 5150 course description. "Microcomputer operations and educational applications (3) Operating procedures and programming techniques. Hands-on experience in operating common microcomputers, writing, debugging, and running educational programs" (U.T. Graduate Catalog, 86-88)

Design of the Study

The research design used for this study was the quasi--experimental pretest-posttest control group design. According to Campbell and Stanley (1963), this design was considered to be one of the better quasi-experimental designs for use with experimental and control groups. The design is represented by the following diagram:

TABLE 1

Student Enrollment in TAE 5150 and Class Assignment to Control or Experimental Group During Spring and Summer Quarters, 1987

Term	Section	Group	N	Percent
Spring, 1987	73980	Experimental	14	63.6
	73993	Control	8	36.4
		TOTAL	22	100.0
Summer, 1987	40327	Experimental	11	57.8
	40330	Control	8	42.2
		TOTAL	19	100.0

O1	X	O2	O3	O4
O1		O2	O3	O4

Notations:

O1 = A written pretest that measured students' initial knowledge of BASIC programming language and demographic survey.

X = The treatment (supplementary aid examples) on BASIC programming instruction supplied to the experimental group.

O2 = A written posttest that measured students' knowledge of BASIC programming language administered in the sixth week of experiment. It was the same as the pretest.

O3 = The final exam, which measured students' knowledge of BASIC programming language at the ninth week of the quarter. It was similar in format to the knowledge test.

O4 = The term project (final program), which measure student programming skills.

Due to the limitation of the number of computer-programming courses taught, and the number of students enrolled in these courses, a large sample of students and courses could not be obtained. The two available classes (control group and experimental group) were selected and served as the sample for the study. Since, this study could prove to have weak external validity, replication of the study was conducted. Two TAE 5150 classes taught during the first session of the Summer quarter, 1987, served as the sample for the replication study.

Statistical Analysis

The study was conducted using two groups that were not related other than being participants in two sections of the same course. Data that were used were the pretest to posttest gain scores, the final test scores, and the scores of the term project. The data from all the three parts (1,2, and 3) of the pretest to the three parts of the posttest gain scores, were also collected and analyzed. The teaching method served as an independent variable.

The dependent variables considered in this study were: Pretest-to-posttest gain scores, final-examination scores, and scores on the final projects. The t-test was chosen to test the null hypothesis.

Delimitations

This study was delimited to examining students' achievement and programming skills. The supplementary aid examples were developed to be used as a supplement to the experimental teaching method. In addition, this study examined the impact of two instructional methods for teaching the BASIC programming language.

The population of the study was delimited to all the students enrolled in the two sections of the course TAE 5150 at the University of Tennessee during the Spring quarter, 1987, and the 19 students enrolled in two sections of the same course TAE 5150 during the first session of the Summer quarter, 1987.

Limitations

The inability to random sample a large student population attending computer-programming language course(s) was a severe limitation. Although, the study was replicated, the findings of the study were limited in generalizability to the population of the students enrolled in the TAE 5150 at the University of Tennessee.

Definition of Terms

BASIC: An acronym for Beginners All-purpose Symbolic Instruction Code, a computer programming language.

Lecture and demonstration method of instruction: A modified form of traditional lecture teaching style combined with demonstrations of commands, statements,

and routines of the BASIC language on the microcomputer. Feedback from students during instruction determine the pace of instruction.

Microcomputer: A computer whose main central processing unit (CPU) is a microprocessor. This computer, usually desktop in size, includes the main unit with CPU, memory and disk drives, a keyboard for entering data, and a TV-like monitor for displaying information.

Supplementary aid examples: Structured programs that were developed by the researcher to reflect techniques of good programming, serve as a concrete model, and provide opportunities for students to have more hands-on experience.

CHAPTER II

LITERATURE REVIEW

The review of the related literature for this study was conducted using three different search techniques:

1. Computerized search of several data bases that related to this research; (a) The Computer Database, (b) Microcomputer Index, (c) Educational Resources Information Center (ERIC), (d) Dissertation Abstract International, and (d) National Technical Information Service (NTIS)

2. A manual search through available books and journals at The University of Tennessee Libraries.

3. Personal contacts, especially materials supplied by the researcher's professors.

Although the search was extensive, there was minimal related literature in the area of teaching computer-programming languages. This may be attributed to the short history in the area of teaching computer-programming languages using the microcomputer.

Justification for the Study

> In spite of tremendous achievement in hardware development and the advances in producing educational software, there has been comparatively little basic research on how to teach computer programming (Mayer, 1982, p. 6).

In the primitive research already done, there were contradictory conclusions concerning the best methods of teaching computer programming and how children and adults learn computer programming. Computer programming has not enjoyed a long

history of instructional research like other fields such as mathematics and science (Mayer, 1984).

If educators are to determine how students learn computer programming, the best methods to teach computer programming, and how to use the computer efficiently and effectively, more research has to be conducted to expand the body of knowledge in the area of computer programming. As stated by Borg and Gall (1983) in Wiggins (1984, p. 9):

> The major reason for educational research is to develop new knowledge about teaching and learning and administration. The new knowledge is valuable because it will lead eventually to the improvement of educational practice (Borg and Gall, 1983, p. 4).

Most of the existing educational research on teaching about computers and computer programming has dealt with computer-aided instruction versus traditional methods of instruction (Wiggins, 1984). Research in the area of hands-on experience, required assignments, and structured approaches to teaching and learning computer programming are minimal. Therefore, research from other related disciplines was examined.

Survey of Recent Research

Problem Solving

Computer programming has been incorporated into school curricula at all levels, in part due to the belief by many educators that computer programming is an important skill for all children in today's technological society (Kurland et al., 1984). There are many claims by many researchers that cognitive skills are inherent in computer programming. However, Kurland et al. (1984) have stated that:

> There is little evidence that current approaches to teaching programming brings students to the level of programming competence

> needed to develop general problem-solving skills, or to develop a model of computer functioning that would enable them to write useful programs (Kurland et al., 1984, p. 1).

On the question of what computer programming can do for students, there is clear agreement that it can help students with problem-solving activities in the classroom (Mandinach and Linn, 1986). Means (1985) acknowledged that in teaching a computer language, students should be encouraged to develop or improve their problem solving skills rather than concentrate on teaching syntax of the programming language only.

Mathematics

Many educators have found through empirical research that mathematical performance is related to or associated with computer-programming skills. Sauter (1986) stated the following:

> ... studies such as Paterson and Howe[1] and Butcher and Muth[2] have demonstrated a relationship between high school mathematics/science performance and college computer science performance. Similarly, Kovalina et al.[3] related high school and college mathematics exposure to performance in a particular computer science course and Alspaugh[4] found mathematics aptitude to be related to programming proficiency (Sauter, 1986, p. 299).

However, there is speculation that mathematics ability is not the only predictor of programming skill. It is believed that programming ability is associated with language proficiency and aptitude (Sauter, 1986).

There is a notion among many psychologists today that computer programming can have impact on children and can help them to develop general high level skill that is useful in subjects such as mathematics (Kurland et al., 1984).

Programming Skills

Nowaczyk (1983) conducted research to identify what factors might predict success in computer programming. His sample consisted of 286 college students from

three computer courses (160 students from introductory programming, 60 from COBOL programming, and 66 from senior-level programming). A 30-minute question-answering and problem-solving test was administered. The cognitive factors (past academic performance and problem-solving ability) were assessed through self-reported grades and seven math and logic problems. Personality factors such as computer anxiety and locus of control were assessed by items from the Fenneman and Sherman Mathematics Attitude Scale.

Results showed that (a) past academic performance in mathematics and English courses, (b) amount of previous computer experience, (c) expected grade in the course, and (d) performance on selected logic and algebraic word problems were significantly correlated to course performance.

Programming

Mayer (1982) conducted a project that provided new information on the process of how novices learn computer programming and how to help novices to become more productive users of computers. Mayer concluded:

1. A Concrete model can have a strong effect on the encoding procedure and the use of new information by novices.

2. Students seem to assimilate each new statement in a computer-programming language to their own image when they are exposed to appropriate models.

Kurland et al. (1986, p. 429) conducted a one-year long experimental study of high-school students learning computer programming. The purpose of the study was to examine three points (a) The impact of programming on particular mathematical and reasoning ability, (b) Which cognitive skills or abilities best predict programming ability, and (c) what students actually understand about programming after two years of high school study?

The study concluded that even after two years of studying programming, students' understanding of programming was primitive and "...that programming experience doesn't appear to transfer to other domains which share analogous formal properties." (p. 429).

Kurland, et al. (1986, p. 429) concluded that:

> We need to more closely study the pedagogy of programming and how expertise can be better attained before we prematurely go looking for significant and wide-reaching transfer effects from programming.

Wiggins (1984) conducted research to identify factors that affect student achievement and attitudes in a BASIC computer-programming course. Two teaching methods were used, computer-assisted instruction and traditional instruction, on 103 college students in a BASIC computer-programming language course during two successive semesters. A pretest and a posttest were given to all students in both, the experimental and control groups to measure students' knowledge of the BASIC computer-programming language. A pretest and posttest were administered to both groups to measure their attitudes toward computer programming.

Wiggins (1984, p. 76) found the following:

1. There was no significant difference in either posttest attitude scores or posttest knowledge scores when the students were grouped by teaching method.

2. There was a significant difference in posttest knowledge scores when subjects were grouped by; pretest, attitude scores, semester of vocational agriculture, subjects in which students achieved their highest and lowest grade in high school, average secondary and post-secondary math grade, student classification, student major, videogame experience, or the person who most influenced the student to take the course.

3. There was a significant difference in posttest attitude scores when subjects were grouped by their typing ability or computer experience.

Tsai and Pohl (1978) conducted a study to show the differences in student learning achievement. Using a college-level computer-programming course, three teaching and learning environments (a) lecture, (b) computer aided instruction, and (c) lecture supplemented by computer aided instruction were studied. This study failed to show any significant difference in students' achievement among the three teaching methods when grouped by either homework scores or term projects. However, a significant difference was found when students' achievement was measured by either hour-long quiz scores or final-examination scores.

Tsai and Pohl (1978, p. 70) stated that:

> A CAI teaching/learning environment--either by itself or as a supplement to lecture instruction--is at least equal to and quite possibly is more effective than the traditional lecture format for college students learning computer-programming.

Using a quasi-experimental study on three intact classes enrolled in a computer-concepts course, Dershimer (1979) compared the effect of three methods of instruction to three groups. One class received the program (an interactive computer program used as an instructional aid in teaching BASIC flow charting techniques) as an individualized method of instruction. The second received the program as a supplement to the lecture methods of instruction, while the third class, which served as the control group, received the conventional lecture method of instruction.

Dershimer, evaluated nine hypotheses using the analysis of covariance at the .05 level of significance. None of the nine hypotheses were rejected. It was concluded that any of three methods used to teach BASIC flow charting using measured achievement in flow charting and attitudes toward flow charting, produced similar results. There were no significant differences among the three methods. It was recommended by Dershimer that the interactive program to aid in BASIC flow charting techniques could be used as an individualized method or combined with the normal classroom instruction.

Debugging

Carver and Klahr (1986) conducted research on nine children, aged seven to nine years. These children received twenty-four hours of LOGO training over a three-week period. A new model proposed by Carve and Klahr was used to assess acquisition of debugging skills by children.

Results of the research showed that students learned the editing and command generation skills prerequisite to debugging but were not able to interpret commands and use clues to identify and locate the bugs.

Du Boulay (1986) concluded that programming was a complex skill to learn. This was especially true in languages such as BASIC and Pascal. They were designed for the novice, but have unpredictable traps for beginners.

The need to specify, develop, test, and debug a program using whatever tools students have available to them, is one of the issues of difficulties in learning computer programming (Du Boulay, 1986).

Favaro (1986) thought programming was a difficult skill that took a long time to master. It required study, practice, and training. According to Kurland (1984, p. 45) "Many students fail to achieve even a modest understanding after one or two programming courses."

Instructional Methods

Despite the high demand and need for computer literacy, there seem to be an agreement among educators that there are several limitations confronting current instructional methods for learning the computer-programming languages. Dalbey and Linn (1986, p. 90) based on their experience, listed several important limitations that are confronting programming instruction. They are as follows:

1. Students in computer courses, have limited access to computers.

2. Teachers' experience and expectations are limited. Teachers are exposed to limited training in the computer subjects that they are assigned to teach.

3. The curriculum materials available for pre-college computer-programming courses are primitive and often fail to lead students to a modest cognitive accomplishment.

4. Choice of programming languages used, are not the most appropriate language for learning an acceptable design skills.

Perkins et al. (1986, p. 53) stated that:

> The relatively undeveloped pedagogy of computer programming instruction falls short on two fronts. It does not teach directly the sort of schemata a skilled programmer needs. Moreover, it does not guide students in ways that foster bootstrap learning of those schemata.

Educators believed that computers and their related technology were changing the way teaching was being conducted at all levels of education and especially at the college level. McKeen (1986) noted that offering new approaches to teaching and learning environment, increased the ability to tailor instructional packages to specific tasks. Although, McKeen warned against new approaches in teaching. McKeen (1986, p.369) said that:

> With the promise of new approaches also come the pitfalls. Care must be exercised in adopting these technology-inspired methods. We must be careful to proceed in accordance with our hard-earned knowledge concerning teaching and learning--from theoretical approach as well as from an experimental approach.

The idea of integrating computer education into the school curriculum has raised a problem for teacher educators, where they have to use computers and teach about them in their classrooms. Russell (1984) in Wiggins (1984, p.15) stated that:

> To operate a microcomputer effectively, instructors must acquire some basic skills in computer literacy. For example, instructors must know whether a computer software program is compatible with a particular type of hardware. They must have an understanding of the microcomputer commands to execute a computer program and to

> perform basic operations such as printing, data storage and retrieval, and running pre-programmed software (Russell, 1984, p. 2).

Lecture and demonstration methods of instruction. The most popular method of instruction used in classroom is the lecture-demonestration method (Jumpeter, 1985). Other teaching methods discussed included the lecture method, computer-assisted instruction, computer-managed instruction, auto-tutorial instruction, and the personalized system of instruction. Jumpeter (1985, p.114) defined the lecture and demonstration method as:

> For the lecture-demonstration method, the instructor is the prime participant, while the student's activities and progress are noticeable only when taking quizes or examinations or participating in classroom discussions.

The lecture and demonstration method is popular in teaching technical subjects such as Physics, Chemistry, Mathematics, Astronomy, Medicin, and computer programming. It is used where concepts and skills have to be demonstrated to provide some concrete examples to enhance students' understanding. Bodner (1985) acknowledged that there had been a recent high resurgence of interest in lecture demonstrations. Bodner listed several factors to support the importance of the lecture demonstrations; (a) they are fun, (b) they grab the students attentions, (c) students seem to like them, (d) they provide a break that help students recover from the deluge of information in a typical lecture, and (f) they can provide examples of abstract concepts.

Connolly (1985, p. 529) stated that " Introducing students to the concept of structured programming via lecture on the merits and benefits of top-down design and proper documentation is not enough." Connolly suggested that:

> Students should have the opportunity and requirement to work in an environment that requires developing programs using structured techniques, and their work should be evaluated in a manner that both encourages and rewards proper use. Such a learning environment not only teaches the precepts of structured programming, it aids the development of programming habits that are essential.

Educators believe that demonstrations during the lecture method is an invaluable aid to teaching a skill. In teaching a computer-programming language, the instructor uses demonstrations, not only to present and teach a programming concept, but also to show how to perform an operation in a particular way. It is used to teach students why some computer-programming concepts, statement, and commands work in a particular way.

Anxiety

Many educators believe that continuous technological advances in the area of computer technology and knowledge have contributed to numerous changes on the attitude of today's technological society. These changes are believed to have created the feeling of anxiety, stress, and humiliation among computer users.

According to Bennett (1985), fear and apprehension caused by being exposed to electronic computer equipments is called computer anxiety. Bennett also said, that computer anxiety ranges from the feelings of stress and discomfort, when exposed to a computer, to rage and anger. These ranges of anxiety have a major impact on novices learning computer programming.

Bennett (1985) listed several causes and cures of computer anxiety. Some of the causes suggested and listed by Bennett were:

1. Some computer users doubt that they can operate or have the ability to operate a computer because of their poor self image.

2. There are difficulties understanding computer manuals and books, especially for people with reading problems.

3. The availability of computer equipment and well trained teachers to teach the general public is inadequate.

4. There is no standardization of computer curriculum and training for teachers and their students.

5. The existing anxiety of teachers can be passed down to their students.

6. People resist changing their habits. Using the computer require some changes and development of a new behavior pattern.

Some of the cures suggested and listed by Bennett were as follow:

> 1) Widespread computer training, taught in simple, understandable language should be offered to people of all ages. 2) There should be increased opportunities for "hand on" training at schools, workshops, etc. 3) Companies should strive to produce computer hardware that is easy to operate without requiring much prior knowledge or training. . . . 5) There should be increased emphasis on producing "userfriendly" software. 6) Keyboard training should offered early in the lower grades to establish correct habits while children are first learning to use a computer. . . . 10) Computer texts should incorporate motivational materials that reassure, encourage, and guide, entertain, and direct readers into learning to use a computer. 11) Computer methodology courses and texts for teachers should be developed and curriculum standardized (p. 24).

Powers (1973) conducted a study to investigate the following: (a) the relationship between user anxiety and the utilization of computer facilities, (b) the effect of hands-on computer experience on anxiety level of computer operators, and (c) the effect of prior computer exposure on anxiety level. The sample of the study consisted of volunteer college students who then were assigned to either one of three groups based on their prior computer experience.

Powers stated that, "Results indicated that regardless of previous exposure, subjects experienced anxiety, as reflected by physiological changes, during the initial parts of the computer task, but that continued "hands on" manipulation reduced anxiety below the pre-exposure level."

CHAPTER III

PROCEDURES AND METHODOLOGY

Procedure

Programs and routines for the BASIC (Beginners All-Purpose Symbolic Instruction Code) computer-programming language on personal computers were developed by the researcher and provided to the experimental group. They served as supplementary aid examples and the treatment for the study. The supplementary exercises were activities used to engage the students directly with the computer. These structured programs were developed to provide opportunities for students to have more hands-on experience, increase the students' understanding of a computer-programming language, and enhance their ability to read the code in computer programs.

Each supplementary exercise program written used a small number of statements or commands, designed to illustrate one or more concept(s). Only students in the experimental group received these exercise programs as the experimental treatment. The concepts, statements, and commands covered in all of the programs were taught by the same professor to all classes and also were covered in the text book. Both the experimental and control group had equal access to the microcomputer lab. Except for the experimental treatment, the same assistance was given as requested by the students, with no distinction between groups.

The target population of this study was all the students attending the Technological and Adult Education (TAE) BASIC programming language courses. The accessible population of this study consisted of all students registered for the two sections of the introductory microcomputer-programming course, TAE 5150, during the Spring, 1987 quarter, in the TAE Department at the University of Tennessee. This type of sample

is often called "accidental" or "convenient" sample. One of the two sections of the course (TAE 5150) was assigned as the experimental group and the other as the control group by a simple coin toss. The same sample type and techniques were used in the replication study during the first session of the Summer quarter, 1987.

A written pretest was administered to each group in the same classroom under the same environmental conditions (e.g., lighting, seating, etc.) at each first-class meeting, to determine the student initial knowledge level. The pretest and posttest were exactly the same and were called the knowledge test. The exam used was the regular midterm exam administered by the same professor. It had been validated and had been shown to be reliable over 6 quarters of use. The knowledge test consisted of three parts, (a) true and false questions, (b) a program with several errors that needed to be debugged, and (c) an error-free program that needed to be traced to determine which line(s), variables, or statement(s) performed what function. Test questions covered materials taught during the first five-weeks.

Experimental Treatments

The experimental group was determined to be the Tuesday evening class by coin toss. Supplementary aid examples were developed by the researcher and used by students in the experimental group. Each supplementary aid example was designed to present one or more concept(s) of the BASIC programming language and consisted of four parts: (a) objectives, (b) statements, (c) program example, and (d) instruction. The objectives section listed the items and concepts to be learned. The statements section presented all the statements that were used in the program. Each statement's or function's application was described using remarks in the same line. The program section contained a program example that presented all the concepts to be taught. Finally, the instruction section, gave

direction to students concerning what needed to be done. All supplementary aid examples' materials are included in Appendix A.

All the concepts, commands, functions, and statements illustrated in the supplementary aid examples were covered in the same sequence in the lecture and demonstration periods for both groups. All this material also was covered in the textbook used for this course.

Students in the experimental group were required to key in all the supplementary aid examples and make required changes. They practiced the structured supplementary aid examples that gave them detailed objectives, program examples, statements and commands illustrations, and direction of what needed to be done. A printed copy of the final program listing and its output was turned in to the instructor. All the returned assignments were then checked by the researcher and returned to the student.

Control Group

The control group had the same equal access to the microcomputer laboratory as did the experimental group. Both groups (experimental and control) received the lecture and demonstration instruction and used the same book for the course. However, The experimental group received the treatment exercises.

The researcher attended both sections on an equal basis and observed the instruction in both sections. The instructor attempted to keep the lecture and demonstration sessions the same for both sections. During the lab time, the students in the control group were asked to practice some examples from the book.

Instruments

Four instruments were developed by the researcher and the course instructor for the study: (a) the information sheet (demographic survey), (b) the knowledge test (pretest and posttest), (c) the final examination, and (d) Program check list.

Information Sheet

The demographic survey was developed by the researcher and was titled "Information Sheet." This questionnaire was designed to collect specific demographic data that would contribute to the control of the research and secure information on important personal differences that might have contributed to achievement if any.

This instrument consisted of 11 items: social security number (SSN), program major, age, sex, present study classification, number of computer-programming language courses taken prior to the course, length of formal instruction with micro-computers, length of formal instruction with mainframe computers, number of math courses completed with a 'C' grade or better in high school and college, present undergraduate GPA, and factors that most influenced them to attend the course TAE 5150. It was administered to both the experimental and control groups during the first class meeting of each section and just before the pretest. A copy is included in Appendix B.

Knowledge Test

The knowledge test served as the pretest and the posttest. It consisted of three parts:

Part 1. Thirty three true-false questions about the BASIC programming language and its relationship to the microcomputer software and hardware.

Part 2. A program written in BASIC that contained 10 errors. Students' scores in this section were based on two criteria; (a) identifying the error, and (b) correcting the error. A point was given for identifying the error and a point for correcting it.

Part 3. An error free BASIC program about which 16 questions were asked concerning the flow of the program and what line, variable, or statement performed what action.

This knowledge test was administered as a pretest to both the experimental and control groups during the first class session, following the demographic survey. The same test was administered as a posttest to both groups during the sixth session. A copy of this test can be found in Appendix C.

Final Examination

The final examination was developed to test students' overall knowledge of all general concepts and principles covered during the course. It was similar to the knowledge test in its three-part format, but covered the content of the entire course. It was used to determine if any changes in students' performance would occur between the end of the treatment period and the end of the course.

Term Project

The final project tested the students' ability to write a structured program and the ability to demonstrate knowledge and understanding of the BASIC language concepts and principles. Each student of both groups was required to write a computer-aided instruction (CAI) type comprehensive program that utilized the following:

1. Menu driven, CAI type programs.
2. The specific functions and statements covered during lectures and required in the program.

3. Good documentation using REM statements.

A program checklist was develop by the researcher to determine a grade on the final project for each individual student. The instrument is shown in Figure 1.

Analysis of Data

In this study, the dependent variables for the seven hypotheses were scores on the pretest, posttest, final exam, and the final project. The independent variable for each of the seven hypotheses was teaching method. The data were collected and entered in a text file on the University of Tennessee Computer Center (UTCC) VAX system. Statistical Analysis System (SAS) programs were written to analyze the data. Procedures: PRINT, FREQuency, TABLES, CORRelation, TTEST, and ANOVA were used to present an illustration of tables, counts, and diagrams of the data. The t-test was used to test the null hypotheses. The level of significant difference used in the analysis of data in this study was .05, unless specifically stated otherwise.

Summary

Two sections of the TAE 5150 course at the University of Tennessee, taught during the Spring quarter, 1987, were used to serve as the experimental and control groups for the initial study. The study was replicated on two sections of the TAE 5150 course conducted during the first session of the Summer quarter,1987. A quasi-experimental pretest-posttest control group design was used as the design for the study. The objectives of this study were to determine if the instructional method used, supplemented with supplementary aid examples would produce a significant difference in the achievement scores and programming skills between the two groups. Fifteen supplementary aid examples were developed and distributed to the experimental group only, and served as the

treatment for the study. Four instruments to collect needed data were developed by the researcher and the course instructor. The data were collected and then analyzed using the SAS system on the VAX system at the Computer Center, The University of Tennessee.

CHAPTER IV

ANALYSIS OF DATA AND FINDINGS

The purpose of this study was to determine, through experimentation, if a structured, conceptual method of instruction, combined with assigned structured supplementary examples of computer-program routines, would produce higher test scores and better written programs than the traditional lecture and demonstration methods of instruction. Comparisons and analysis were made of students' tests and final-projects scores between the experimental group and control groups. This information then was used to test the seven null hypotheses. The demographic data collected were analyzed to determine any possible effects on the experimental data.

The Initial Study (Spring, 1987)

Demographic Characteristics

Twenty-two subjects were enrolled in the two sections of the course TAE 5150 during the Spring quarter, 1987. Both sections were night classes, one on Tuesday and the other on Thursday. The Tuesday-night section, with 14 students, was chosen by coin flip to be the experimental group. Eight students were enrolled in the Thursday night section, which was the control group. (See Table 1, page 7). Data on the eight demographic characteristics included in the demographic survey; age, sex, study classification, number of programming courses attended, formal instruction using micro-computers and mainframe computers, number of math courses, and their GPA were collected and analyzed. The results of the demographic data analysis can be found in Table 2.

TABLE 2

Demographic Characteristics of the Students Enrolled in TAE 5150 During Spring, 1987

Item	Variable	Value	Experimental Group (N 14) Frequency	Experimental Group (N 14) Percent	Control Group (N 8) Frequency	Control Group (N 8) Percent
1.	Age					
	Less than 25	1	2	14.3	4	50.0
	25 or older	2	12	85.7	4	50.0
2.	Sex					
	Female	1	8	57.1	7	87.5
	Male	2	6	42.9	1	12.5
3.	Present Study Classification					
	Graduate	1	12	85.7	7	87.5
	Undergraduate	2	1	7.1	1	12.5
	Special	3	1	7.1	0	0.0
4.	Courses of Programming Language					
	None	0	10	71.4	7	87.5
	1 Course	1	3	21.4	0	0.0
	2 Courses	2	0	0.0	0	0.0
	3 Courses	3	1	7.1	1	12.5
	More than three	4	0	0.0	0	0.0
5.	Have Had Microcomputer Instruction					
	None	0	12	85.7	6	75.0
	Less than 6 months	1	1	7.1	2	25.0
	6 months-a year	2	1	7.1	0	0.0
	More than a year	3	0	0.0	0	0.0

TABLE 2 (Continued)

Item	Variable	Value	Experimental Group (N 14) Frequency	Experimental Group (N 14) Percent	Control Group (N 8) Frequency	Control Group (N 8) Percent
6.	Have Had Mainframe Instruction					
	None	0	11	78.6	7	87.5
	Less than 6 months	1	2	14.3	1	12.5
	6 months-a year	2	1	7.1	0	0.0
	More than a year	3	0	0.0	0	0.0
7.	Math Courses Completed With a 'C' or Better					
	1	1	1	7.1	1	12.5
	2	2	1	7.1	0	0.0
	3	3	4	28.6	1	12.5
	4	4	2	14.3	1	12.5
	5	5	2	14.3	1	12.5
	6	6	0	0.0	0	0.0
	7	7	1	7.1	1	12.5
	8	8	1	7.1	2	25.5
	9	9	2	14.3	1	12.5
8.	GPA					
	Below 2.00	1	0	0.0	0	0.0
	2.00 - 2.59	2	2	14.3	0	0.0
	2.60 - 3.00	3	5	35.7	4	50.0
	3.01 - 3.50	4	5	35.7	3	37.5
	Above 3.50	5	2	14.3	1	12.5
9.	Factors Influenced to Take the Course					
	Required	1	2	14.3	3	37.5
	Personal Interest	2	6	42.9	4	50.0
	Influence By Others	3	3	21.4	1	12.5
	To Learn BASIC	4	1	7.1	0	0.0
	For Experience	5	2	14.3	0	0.0
	Other	0	0	0.0	0	0.0

Age. Of the 14 subjects enrolled in the experimental group, 85.7 percent (12), were 25 years of age or older. Only 14.3 percent (2), were younger than 25 years of age.

Of the 8 students enrolled in the control group, 50.0 percent (4), were 25 years or older, while the other 50.0 percent (4) were younger than 25 years of age.

Sex. Of those enrolled in the experimental group, 57.1 percent (8), were female and 42.9 percent (6), were male.

In the control group, 87.5 percent (7), were female and only 12.5 percent (1), was male.

Study Classification. In the experimental group, 85.7 percent (12), were graduate students, 7.1 percent (1), was undergraduate, and 1 subject was classified as special, not working on a degree.

In the control group, 87.5 percent (7), were graduate students and 12.5 percent (1), was undergraduate.

Programming-Language Courses. Prior to this course, 71.4 percent (10), in the experimental group had attended no prior programming language courses, 21.4 percent (3), had completed one such course, and 7.1 percent (1), had completed three courses.

In the control group, 87.5 percent (7), had no programming-language courses prior to this study and only 12.5 percent (1), had completed three courses.

Micro-Computer. When subjects were asked how much formal instruction they had had using micro-computers, 85.7 percent (12), of the experimental group received no

prior formal instruction, 7.1 percent (1), had less than 6 months, and 7.1 percent (1), had instruction using micro-computer for a period of 6 months to a year.

In the control group, 75.0 percent (6) received no prior formal instruction while 25.0 percent (2), had received less than six months instruction on the use of microcomputers.

Mainframe computers. The previous question was repeated to see how many students had had formal instruction using mainframe computers. In the experimental group, 78.6 percent (11) had received no prior instruction using the mainframe, 14.3 percent (12), had less than sixth months, and only 7.1 percent (1), had received instruction for a period of six months to a year.

Only 12.5 percent (1), in the control group had received some formal instruction using the mainframe for a period of less than six months, while 87.5 percent (7), had no prior experience.

Math. When students were asked to check all the mathematics courses they had completed with a grade of 'C' or better prior to college and in college, the mean number of courses checked by the experimental group was 4.71, and only 3.14 for the control group.

GPA. The reported GPA for the experimental group was between 2.60 and 3.00. In the control group, the average was slightly higher, but still was between 2.60 and 3.00.

Pretest, Posttest, Final exam, and Final Project Scores

The knowledge test (pretest and posttest) was administered to both groups at the first class meeting and on the sixth week of the quarter. The final exam was administered

in the ninth week of the quarter. The final project was collected and was evaluated using the program checklist shown in Figure 1. All of the tests and the final project were graded by the researcher and the data were tabulated and analyzed using the Statistical Analysis System (SAS) on the VAX computer system. Each of the tests (the knowledge test and final test) was composed of three parts; part 1, the general conceptual true and false questions, part 2, errors and bugs location and correction, and part 3, the identification of specific variables, statements, functions, and routines). Scores for each individual student, in all the three parts of the pretest, posttest, and the final exam were tabulated. Scores on the final project also were determined. The raw scores for the pretest and posttest can be found in Table 3. The raw scores of the final test and final project can be found in Table 4

Analysis of The Null Hypothesis

Hypothesis 1. The null hypothesis Hø1 stated that there was no significant difference in the performance of students who received supplementary aid examples, as reflected by the total pretest-posttest mean-gain scores, and those who received the traditional lecture and demonstration instruction only.

The experimental group did perform better as reflected by the pretest-posttest mean-gain scores (M=64.57) than did the control group (M=45.25), with $t = 2.22$, $p<.05$. Based on this result, the null hypothesis was rejected. Presented in Table 5 are the t-test results for the difference of the two means of pretest-posttest gain scores between the two groups.

Hypothesis 2. The null hypothesis Hø2 stated that the general conceptual knowledge of student who received supplementary aid examples was not significantly different from those who received the traditional lecture and demonstration instruction only.

PROGRAM CHECKLIST

Negative Points

FUNCTIONAL PROBLEMS

1. Incomplete Program	(-60) ____
2. Run-time(execution) Errors But depends on severity)	(-40) ____

DOCUMENTATION PROBLEMS

1. Variables are not named appropriately, according to the quantities they represent	(-10) ____
2. Inadequate header Documentation (Detailed program and variable descriptions)	(-10) ____
3. Inadequate body documentation (REMarks)	(-10) ____
4. Remarks superfluous/repetitious/inappropriate	(-10) ____

STRUCTURE PROBLEMS

1. Inadequate/incomplete explicit variable declarations and initializations	(-10) ____
2. Excessive GOTO Statements (Unstructured program)	(-15) ____
3. Program logic vague,round-about and difficult to trace even though program seems to give the right results	(-10) ____
4. Program is not written in a small modules (SUBROUTINES)	(-10) ____
5. Not a MENU driven program	(-10) ____

APPEARANCE PROBLEMS

1. Program not indented clearly/well (Documentation)	(-15) ____
2. Output sloppy/inadequate	(-20) ____

Positive (Bonus) Points

1. Exceptional appearance of output	(+4) ____
2. Exceptional appearance of program	(+4) ____
3. Extra features provided	(+8) ____

Total points deducted	= _____
Total points added	= _____
Total points (100 +)	= _____

FIGURE 1. Program Checklist

TABLE 3

Pretest and Posttest Scores for Student in The Initial Study, Spring, 1987

	Experimental Group (N 14)				Control Group (N 8)			
	Score	Frequency	Percent	Mean	Score	Frequency	Percent	Mean
Pretest								
	2	1	7.1		6	1	12.5	
	4	1	7.16		36	1	12.5	
	6	1	7.1		45	1	12.5	
	10	2	14.3		46	1	12.5	
	12	1	7.1		50	1	12.5	
	20	1	7.1		55	2	25.0	
	32	1	7.1		56	1	12.5	
	34	1	7.1					
	38	1	7.1					
	42	1	7.1					
	44	1	7.1					
	58	1	7.1					
	60	1	7.1					
Total		14	100.0	26.57		8	100.0	43.6
Posttest								
	52	1	7.1		70	1	12.5	
	68	2	14.3		84	1	12.5	
	72	1	7.1		86	2	25.0	
	78	1	7.1		87	1	12.5	
	94	1	7.1		94	2	25.0	
	98	1	7.1		110	1	12.5	
	100	2	14.3					
	108	3	21.4					
	110	1	7.1					
	112	1	7.1					
Total		14	100.0	91.14		8	100.0	88.87

TABLE 4

The Final Project and Final Exam Raw Scores For Students in The Initial Study, Spring, 1987

	Experimental Group (N 14)				Control Group (N 8)			
	Score	Frequency	Percent	Mean	Score	Frequency	Percent	Mean
Final Project								
	25	1	7.1		45	2	25.0	
	40	3	21.4		50	1	12.5	
	59	1	7.1		65	3	37.5	
	60	1	7.1		74	1	12.5	
	65	1	7.1		80	1	12.5	
	77	1	7.1					
	80	2	14.3					
	90	1	7.1					
	92	1	7.1					
	106	1	7.1					
	108	1	7.1					
Total		14	100.0	68.71		8	100.0	61.12
Final Exam								
	68	1	7.1		80	1	12.5	
	72	2	14.3		82	1	12.5	
	79	1	7.1		85	1	12.5	
	87	1	7.1		95	1	12.5	
	97	1	7.1		96	2	25.0	
	99	1	7.1		97	1	12.5	
	101	1	7.1		101	1	12.5	
	104	1	7.1					
	106	2	14.3					
	110	3	21.4					
Total		14	100.0	94.35		8	100.0	91.5

TABLE 5

t-Test of the Difference in Pretest-Posttest Mean Gain Scores
Between the Two Groups in the Initial Study

Group	N	M	SD	SE	t-value	DF	Probability
Experimental	14	64.57	3.18	6.19	2.22	18.4	0.0319 *
Control	8	45.25	17.25	6.09			

* Significant at the .05 level.

The result of t-test shown in Table 6, $t = 1.49$, $p>.05$, indicated no significant difference in the mean-gain scores from part 1 of the pretest-posttest. At the .05 significance level, null hypothesis Hø2 failed to be rejected

Hypothesis 3. The null hypothesis Hø3 stated there was no significant difference in the debugging ability of students who received supplementary aid examples, as reflected by the mean-gain scores from part 2 of the pretest-posttest, and those who received the traditional lecture and demonstration instruction only.

The result of t-test shown in Table 7, $t = -0.01$, $p>.05$, indicated no significant difference in part 2 mean-gain scores of the pretest-posttest. At the .05 significance level, null hypothesis Hø3 failed to be rejected.

Hypothesis 4. The null hypothesis Hø4 stated there was no significant difference in the ability to read and understand a computer program flow between students who received supplementary aid examples, as reflected by the mean-gain scores from part 3 of the pretest-posttest, and those who received the traditional lecture and demonstration instruction only.

As shown in Table 8, the experimental group reported a significantly greater achievement in part 3 (the ability to identify variables, statements, functions, and routines) with ($M = 22.71$) than did the control group ($M = 13.25$), $t = 3.13$, $p<.05$. It was necessary to reject the null hypothesis.

Hypothesis 5. The null hypothesis Hø5 stated that there was no significant difference in the quality of programs written by students who received Supplementary Aid

TABLE 6

t-Test of the Difference in Part 1 of pretest-posttest Mean Gain Scores
Between the Two Groups in the Initial Study

Group	N	M	SD	SE	t-value	DF	Probability
Experimental	14	32.14	17.49	4.67	1.49	18.1	0.1534
Control	8	22.25	13.32	4.71			

TABLE 7

t-Test of the Difference in Part 2 of pretest-posttest Mean Gain Scores
Between the Two Groups in the Initial Study

Group	N	M	SD	SE	t-value	DF	Probability
Experimental	14	9.71	5.59	1.49	-0.01	16.0	0.9880
Control	8	9.75	5.06	1.79			

TABLE 8

t-Test of the Difference in Part 3 of pretest-posttest Mean Gain Scores
Between the Two Groups in the Initial Study

Group	N	M	SD	SE	t-value	DF	Probability
Experimental	14	22.71	7.98	2.13	3.13	18.2	0.0057 *
Control	8	13.25	6.04	2.13			

* Significant at the .05 level.

Examples, as determined by the mean scores of final projects, and those who received the traditional lecture and demonstration instruction only.

The result of t-test scores in Table 9, $t = 0.90$, $p>.05$, indicated no significant difference in the mean scores of the final project between the two groups. At the .05 significance level, the null hypothesis Hø5 failed to be reject.

Hypothesis 6. The null hypothesis Hø6 stated there was no significant difference in the performance of students who received supplementary aid examples, as reflected by the mean scores of the posttest, and those who received the traditional lecture and demonstration instruction only.

The result of t-test shown in Table 10, indicate no significant difference in the mean scores of the posttest. Based on the t-test, $t = 0.34$, $p>.05$, the null hypothesis Hø6 was not rejected at the .05 significance level.

Hypothesis 7. The null hypothesis Hø7 stated there was no significant difference in the performance of students who received supplementary aid examples, as reflected by the mean scores of the final exam, and those who received the traditional lecture and demonstration instruction only.

The result of t-test shown in Table 11, indicate no significant difference between the mean scores of the final exam. Based on the t -test, $t = 0.56$, $p>.05$, the null hypothesis Hø7 was not rejected at the .05 significance level.

TABLE 9

t-Test of the Difference in Final project Mean Scores
Between the Two Groups in the Initial Study

Group	N	M	SD	SE	t-value	DF	Probability
Experimental	14	68.71	26.04	6.96	0.90	19.8	0.3755
Control	8	61.12	13.15	4.65			

TABLE 10

t-Test of the Difference in Posttest Mean Score Between the Two Groups in the Initial Study

Group	N	M	SD	SE	t-value	DF	Probability
Experimental	14	91.14	19.60	5.24	0.34	20.0	0.7346
Control	8	88.87	11.33	4.00			

TABLE 11

t-Test of the Difference in Final Exam Mean Score
Between the Two Groups in the Initial Study

Group	N	M	SD	SE	t-value	DF	Probability
Experimental	14	94.35	15.57	4.16	0.56	19.9	0.5753
Control	8	91.5	7.91	2.79			

Demographic Characteristics

Nineteen subjects were enrolled in the two sections of the course TAE 5150 first session of the Summer quarter, 1987. Both sections were day classes that met every day of the week for five weeks, one in the morning, and one in the afternoon. The morning section, with 11 students, was chosen by coin toss to be the experimental group. Eight students were enrolled in the afternoon which was the control group. (See Table 1, page 7). Data on the eight demographic characteristics that were included in the demographic survey; age, sex, study classification, number of programming courses attended, formal instruction using micro-computers and mainframe computers, number of math courses, and their GPA, were collected and analyzed. The results of the demographic data analysis can be found in Table 12.

Age. All the 11 subjects in the experimental group were 25 years or older.

Of the 8 students enrolled in the control group, 87.5 percent (7), were 25 years or older, while only 12.5 percent (1),was younger than 25 years of age.

Sex. 36.4 percent (4), enrolled in the experimental group were female and 63.6 percent (7), were male. In the control group, 37.5 percent (3), were female and 62.5 percent (5), were male.

Study Classification. 81.8 percent (9), in the experimental group were graduate students, 9.1 percent (1), was undergraduate, and 9.1 percent (1) was special.

In the control group, 50.0 percent (4), were graduate students, 25.0 percent (2), were undergraduate, and 25.0 percent (2) were special.

TABLE 12

Demographic Characteristics of the Students Enrolled in TAE 5150
During First Session of the Summer Quarter, 1987

Item	Variable	Value	Experimental Group (N 11) Frequency	Experimental Group (N 11) Percent	Control Group (N 8) Frequency	Control Group (N 8) Percent
1.	Age					
	Less than 25	1	0	0	1	12.5
	25 or older	2	11	100.0	7	87.5
2.	Sex					
	Female	1	4	36.4	3	37.5
	Male	2	7	63.6	5	62.5
3.	Present Study Classification					
	Graduate	1	9	81.8	4	50.0
	Undergraduate	2	1	9.1	2	25.0
	Special	3	1	9.1	2	25.0
4.	Courses of Programming Language					
	None	0	9	81.8	5	62.5
	1 Course	1	2	18.2	2	25.0
	2 Courses	2	0	0.0	1	12.5
	3 Courses	3	0	0.0	0	0.0
	More than three	4	0	0.0	0	0.0
5.	Have Had Microcomputer Instruction					
	None	0	9	81.8	4	50.0
	Less than 6 months	1	1	9.1	0	0.0
	6 months-a year	2	0	0.0	4	50.0
	More than a year	3	1	9.1	0	0.0

TABLE 12 (Continued)

Item	Variable	Value	Experimental Group (N 11) Frequency	Experimental Group (N 11) Percent	Control Group (N 8) Frequency	Control Group (N 8) Percent
6.	Have Had Mainframe Instruction					
	None	0	10	90.9	6	75.0
	Less than 6 months	1	0	0.0	0	0.0
	6 months-a year	2	0	0.0	2	25.0
	More than a year	3	1	9.1	0	0.0
7.	Math Courses Completed With a 'C' or Better					
	1	1	1	9.1	1	12.5
	2	2	2	18.2	0	0.0
	3	3	1	9.1	1	12.5
	4	4	1	9.1	2	25.0
	5	5	3	27.3	3	37.5
	6	6	2	18.2	0	0.0
	7	7	1	9.1	1	12.5
	8	8	0	0.0	0	0.0
	9	9	0	0.0	0	0.0
8.	GPA					
	Below 2.00	1	0	0.0	0	0.0
	2.00 - 2.59	2	2	18.2	0	0.0
	2.60 - 3.00	3	4	36.4	3	37.5
	3.01 - 3.50	4	4	36.4	4	50.0
	Above 3.50	5	1	9.1	1	12.5
9.	Factors Influenced to Take the Course					
	Required	1	3	27.3	0	0.0
	Personal Interest	2	0	0.0	0	0.0
	Influence By Others	3	0	0.0	1	12.5
	To Learn BASIC	4	4	36.4	4	50.0
	For Experience	5	4	36.4	1	12.5
	Other	6	0	0.0	0	0.0

Programming-Language Courses. When both groups were asked of how many programming-language courses they have had prior to this course, 81.8 percent (9), in the experimental group, had attended no prior courses, and only 25.0 percent (2), had completed one programming-language course.

In the control group, 62.5 percent (5), had no programming-language courses prior to this study, 25.0 percent (2), had completed one programming-language course, and 12.5 percent (2), had completed two courses.

Micro-Computer. When subjects were asked of how much formal instruction had they had using micro-computers, 81.8 percent (9), of the experimental group received no prior formal instruction, 12.5 percent (1), had less than 6 months, and 12.5 percent (1), had instruction using micro-computer for a period of 6 months to a year.

Fifty percent (4) in the control group received no prior formal instruction and 50.0 percent (4), had received less than six months instruction on the use of microcomputers.

Mainframe computers. The previous question was repeated to see how many students have had formal instruction using mainframe computers. Almost 91.0 percent (10), in the experimental group had received no prior instruction using the mainframe, and only 12.5 percent (1), had received instruction for a period of less than six months.

Only 25.0 percent (2), in the control group had received some formal instruction using the mainframe for a period of six months to a year, while 75.0 percent (6), had no prior experience.

Math. When students were asked to check all the mathematics courses they had completed with a grade of 'C' or better prior to college and in college, The mean number of

courses checked by the experimental group averaged 4.18, while the control group averaged slightly higher at 4.25.

GPA. The reported GPA of the experimental group was between 3.01 and 3.50. In the control group, the average was higher between 3.51 and 4.00.

Pretest, Posttest, Final exam, and Final Project Scores

Data were collected, analyzed, and tabulated using same methods as in the initial study. The raw scores for the pretest and posttest can be found in Table 13. The raw scores for the final test and final project can be found in Table 14.

Analysis of The Null Hypothesis

Hypothesis 1. The null hypothesis Hø1 stated there was no significant difference in the performance of students who received supplementary aid examples, as reflected by the total pretest-posttest mean-gain scores, and those who received the traditional lecture and demonstration instruction only.

The result of t-test shown in Table 15, indicated no significant difference in the mean-gain scores of the pretest-posttest at the .05 level. Based on this result the null hypothesis Hø1 was not rejected.

Hypothesis 2. The null hypothesis Hø2 stated the general conceptual knowledge of student who received supplementary aid examples was not significantly different from those who received the traditional and lecture demonstration instruction only.

From data in Table 16 Comparison of the mean scores in a t-test on the mean-gain scores from part 1 of the pretest-posttest for the experimental and control groups did not

TABLE 13

Pretest and Posttest Scores For Student in the Replication Study, Summer, 1987

	Experimental Group (N 11)				Control Group (N 8)			
	Score	Frequency	Percent	Mean	Score	Frequency	Percent	Mean
Pretest								
	2	1	9.1		33	1	12.5	
	9	1	9.1		40	1	12.5	
	24	1	9.1		44	1	12.5	
	39	1	9.1		48	1	12.5	
	44	2	18.2		52	3	37.5	
	50	1	9.1		70	1	12.5	
	52	1	9.1					
	56	1	9.1					
	58	1	9.1					
	60	1	9.1					
Total		11	100.0	39.81		8	100.0	48.87
Posttest								
	58	1	9.1		79	1	12.5	
	57	1	9.1		82	1	12.5	
	88	1	9.1		86	1	12.5	
	89	1	9.1		88	1	12.5	
	95	1	9.1		98	1	12.5	
	96	1	9.1		105	1	12.5	
	97	1	9.1		106	1	12.5	
	98	1	9.1		110	1	12.5	
	100	1	9.1					
	103	1	9.1					
	115	1	9.1					
Total		14	100.0	95.73		8	100.0	94.25

TABLE 14

The Final Project and Final Exam Raw Scores For Students in the Replication Study, Summer, 1987

	Experimental Group (N 11)				Control Group (N 8)			
	Score	Frequency	Percent	Mean	Score	Frequency	Percent	Mean
Final Project								
	30	1	9.1		15	1	12.5	
	35	1	9.1		45	1	12.5	
	55	3	27.3		55	2	25.0	
	65	1	9.1		70	1	12.5	
	75	1	9.1		78	1	12.5	
	80	3	27.3		88	1	12.5	
	89	1	9.1		90	1	12.5	
Total		14	100.0	63.54		8	100.0	62.00
Final Exam								
	82	1	9.1		89	1	12.5	
	95	1	9.1		96	1	12.5	
	97	1	9.1		99	1	12.5	
	98	1	9.1		103	1	12.5	
	104	1	9.1		105	1	12.5	
	108	2	18.2		106	1	12.5	
	114	1	9.1		107	1	12.5	
	116	1	9.1		118	1	12.5	
	120	1	9.1					
	122	1	9.1					
Total		14	100.0	105.81		8	100.0	102.87

TABLE 15

t-Test of the Difference in Pretest-Posttest Mean Gain Scores
Between the Two Groups in the Replication Study

Group	N	M	SD	SE	t-value	DF	Probability
Experimental	11	55.90	23.26	7.01	1.18	16.9	0.2506
Control	8	45.37	15.27	5.40			

TABLE 16

t-Test of the Difference in Part 1 of pretest-posttest Mean Gain Scores Between the Two Groups in the Replication Study

Group	N	M	SD	SE	t-value	DF	Probability
Experimental	11	22.72	20.81	6.27	0.68	16.1	0.5056
Control	8	17.62	11.58	4.09			

show a significant difference. Based on the result, $t = 0.68$, $p>.05$, the null hypothesis Hø2 could not be rejected.

Hypothesis 3. The null hypothesis Hø3 stated there was no significant difference in the debugging ability of students who received supplementary aid examples, as reflected by the mean-gain scores from part 2 of the pretest-posttest, and those who received traditional lecture and demonstration instruction only.

The result of t-test shown in Table 17, indicated no significant difference in part 1 mean-gain scores of the pretest-posttest. At the .05 significance level, the null hypothesis Hø3 failed to be rejected.

Hypothesis 4. The null hypothesis Hø4 stated there was no significant difference in the ability to read and understand a computer program flow of students who received supplementary aid examples, as reflected by the mean-gain scores from part 3 of the pretest-posttest, and those who received the traditional lecture and demonstration instruction only.

Using data from Table 18 comparison of the mean scores from part 3 of the pretest-posttest, using a t-test equation, indicated no significant difference between the two groups' ability to trace the flow of a program and identify variables and statement content and functions. At the .05 significance level, the null hypothesis Hø4 failed to be rejected.

Hypothesis 5. The null hypothesis Hø5 stated there was no significant difference in the quality of programs written by students who received supplementary aid examples, as determined by the mean scores of the final projects, and those who received the traditional lecture and demonstration instruction only.

TABLE 17

t-Test of the Difference in Part 2 of pretest-posttest Mean Gain Scores
Between the Two Groups in the Replication Study

Group	N	M	SD	SE	t-value	DF	Probability
Experimental	11	12.36	4.43	1.33	0.35	12.8	0.7271
Control	8	11.50	5.70	2.01			

TABLE 18

t-Test of the Difference in Part 3 of pretest-posttest Mean Gain Scores
Between the Two Groups in the Replication Study

Group	N	M	SD	SE	t-value	DF	Probability
Experimental	11	20.81	5.94	1.79	1.52	13.9	0.1500
Control	8	16.25	6.79	2.40			

The result of t-test scores in Table 19 indicate no significant difference in the mean scores of the final project between the two groups. At the .05 significance level, the null hypothesis Hø3 failed to be rejected.

Hypothesis 6. The null hypothesis Hø6 stated there was no significant difference in the performance of students who received supplementary aid examples, as reflected by the mean scores of the posttest, and those who received the traditional lecture and demonstration instruction only.

The result of t-test shown in Table 20, indicate no significant difference in the mean scores of the posttest. Based on the t-test, the null hypothesis Hø6 was not rejected at the .05 significance level.

Hypothesis 7. The null hypothesis Hø7 stated there was no significant difference in the performance of students who received supplementary aid examples, as reflected by the mean scores of the final exam, and those who received traditional lecture and demonstration instruction only.

The result of t-test shown in Table 21, indicate no significant difference between the mean scores of the final exam. Based on the t-test, the null hypothesis Hø7 was not rejected at the .05 significance level.

TABLE 19

t-Test of the Difference in Final project Mean Scores Between the Two Groups in the Replication Study

Group	N	M	SD	SE	t-value	DF	Probability
Experimental	11	63.54	19.37	5.85	0.14	12.7	0.8863
Control	8	62.00	25.00	8.84			

TABLE 20

t-Test of the Difference in Posttest Mean Score Between the Two Groups in the Replication Study

Group	N	M	SD	SE	t-value	DF	Probability
Experimental	11	95.72	8.63	2.60	0.29	12.1	0.7715
Control	8	94.25	11.98	4.23			

TABLE 21

t-Test of the Difference in Final Exam Mean Score Between the Two Groups in the Replication Study

Group	N	M	SD	SE	t-value	DF	Probability
Experimental	11	105.81	12.12	3.65	0.6197	17.0	0.5437
Control	8	102.87	8.5	3.03			

CHAPTER V

SUMMARY, FINDINGS, CONCLUSIONS, AND RECOMMENDATIONS

This chapter summarizes the study, and lists the findings, conclusions, and recommendations of this study for further related research in the area of learning computer programming.

Summary

Due to the lack of related research in the literature in the area of learning and teaching computer programming, this study was undertaken. The purpose of this study was to determine, through experimentation, if a structured, conceptual method of instruction, combined with assigned structured supplementary aid examples of computer-program routines, would produce higher test scores and better written programs than the traditional lecture and demonstration methods of instruction.

A quasi-experimental, pretest-posttest control group, design was used in the study. The population consisted of students enrolled in the BASIC computer-programming courses taught in Technological and Adult Education Department (TAE) in the College of Education at the University of Tennessee. However, the sample for the study was comprised of the 22 students attending the two sections of the course TAE 5150 during the Spring quarter, 1987, the initial study, and the 19 students attended the same course during the First session of the Summer quarter, 1987, the replication study. A t-test was used to test the null hypotheses at a .05 significance level.

Fifteen supplementary aid examples and four instruments were developed for the study; the information sheet (demographic survey), the knowledge test (pretest and posttest), the final examination, and the program evaluation list (checklist).

Findings

1. In the initial study, the experimental group did produce higher mean pretest-posttest gain scores than the control group. However, in the replication study, any apparent difference between the two groups' mean pretest-posttest gain scores was determine to be insignificant.

2. In the initial study, there was determine to be no apparent significant difference between the experimental and control groups' general conceptual knowledge. The same result was found in the replication study.

3. In the initial study, there was found no apparent significant difference between the experimental and control groups' debugging abilities. The replication study produced the same result.

4. In the initial study, the experimental group performed significantly better than the control group in part 3 (tracing and reading a program) of the pretest-posttest. However, in the replication study no significant difference in the students' ability to trace and read a program was found between the two groups.

5. In the initial study, there was determined to be no apparent significant difference between the experimental and control groups' mean final project scores. The same result was found in the replication study.

6. No apparent significant difference was found between the experimental and control groups' mean posttest scores in either the initial study or the replication study.

7. In neither the initial study nor the replication study was there found any apparent significant difference between the experimental and control groups' mean final-exam scores.

Conclusions

The following conclusions are drawn from the summary and findings of the study:

1. A structured method of computer-programming language instruction, combined with assigned structured supplementary aid examples, does not necessarily produce a significantly higher student achievement than the traditional lecture and demonstration method alone.

2. A structured method of computer-programming language instruction, combined with assigned structured supplementary aid examples, does not produce significantly better programming skills than traditional lecture and demonstration method alone.

Recommendations

The study and its findings ought to be a complementary support to enhance the existing and future research in the area of how to teach computer-programming courses, especially teaching the BASIC language. Based on the findings of this study, the following recommendations are recommended:

1. Similar experiments should be conducted with larger sample size, where subjects can be randomly selected, the conclusions of the study could be more conclusive.

2. Further research should be conducted to understand how demographic characteristics effects students' achievement and programming skills.

3. Similar studies should be conducted at the undergraduate level courses to determine the impact of supplementary aid examples on a younger group learning computer programming.

4. Research should be conducted to further determine if students acquire the same skills from structured demonstrations as they do with the structured supplementary aid examples.

5. Research should be conducted to determine what and when appropriate program assignments ought to be assigned to students in computer-programming courses.

6. Instructors of computer programming should be certain to include a strong component of structured programming examples in their demonstrations and adequate hands-on experience with the computer to assure development of the computer-programming skills.

LIST OF REFERENCES

LIST OF REFERENCES

Anderson, R. C., & Faust, G. W. (1973). Educational psychology: The science of instruction & learning. New York, New York: Dodd, Mead & company, Inc.

Bennett, M. (1985). Computer Anxiety: Causes and Cures. Microcomputer in education conference(5th: Arizona State University) Tomorrow's Technology. Computer Science Press, Rockville, MD.

Bodner, M. G. (1985). Lecture demonstration accidents from which we can learn. Journal of Chemical Education, 62(12), 1105-1107.

Bohl, M. (1978). A Guide for programmers. Englewood Cliffs, New Jersey: Prentice-Hall, inc.

Borg, W. R., & Gall, M. D. (1983). Educational research: An introduction (4th ed.). Logman, Inc., New York, New York.

Campbell, T. D., Stanley, C. J. (1963). Experimental and quasi-experimental designs for research. McNally & Company, Chicago, IL.

Carver, S. M., & Klahr,.D. (1986). Assessing children's LOGO debugging skills with a formal model. Journal of Computing Research, 2(1), 487-494.

Chand, D. R. (1978). An automated system for grading BASIC programs. Proceedings of the 16th Annual Convention of the Association for Educational Data Systems (AEDS), May 16-19, 253-264.

Connolly, W. F. (1985). A structured pedagogy for programming courses. In k. Duncan & D. Harris (Eds.), Computer in Education: Proceedings of the IFIP TC 3 4th World Conference, July 29- August 2, 1985, (pp. 529-531). Norfolk, VA.

Dalbey, J., Linn, M. C. (1986). Cognitive consequences of programming: Augmentations to BASIC instruction. Journal of Educational Computing Research, 2(1), 75-93.

Dershimer, W. P. 1979. The development and evaluation of an interactive computer program used as an instructional aid in teaching basic flowcharting techniques. Dissertation Abstract International, 41(8), Feb. 1981.

Du Boulay, Benedict. 1986. Some difficulties learning to program. Journal of Computing Research, 2(1), 57-73.

Ehrlich, K., Abbott, V.,Salter, W., &Soloway, E. (1984). Issues and problems in studies transfer effects of programming. In Kurland, M. D., Development studies of computer programming skills. (ERIC Document Reproduction Service No. ED 257 441)

Favaro, P. J. (1986). Educator's guide to microcomputers and learning. Prentice-Hall, Inc., Englewood Cliffs, New Jersey.

Jumpeter, J. (1985). Personalized system of instruction versus the lecture-demonstration method in a specific area of a college music appreciation course. Journal of research in music education, 33(2), 113-122.

Kurland, M. D., Clement, C., Mawby, R., & Pea R. D. (1984). Mapping the cognitive demands of learning to program. In Kurland, M. D., Development studies of computer programming skills. (ERIC Document Reproduction Service No. ED 257 441).

Kurland, M. D., Pea, R. D., Clement, C., Mawby, R. (1986). Journal of Educational Computing Research, 2(4), 429-455.

Lucas, A., Postma, A., & Thompson, J. C. (1974). A comparative Study of cognitive retention using simulation-gaining as opposed to lecture/discussion technique. Muncie, Indiana. (ERIC Document Reproduction Service No. ED 219 307).

Mandinach, B. E., & Linn, M. C. (1986). The cognitive effects of computer learning environments. Journal of Educational Computing Research, 2(4), 411-423.

Mayer, R. E. (1981). The psychology of how novices learn computer programming. Computer Surveys, March, 121-141.

Mayer, R. E. (1982). Diagnosis and remediation of computer programming skills for creative problem solving. (ERIC Document Reproduction Service No. ED 230 199).

McKeachie, M. J. (1963). Research on teaching at the college level and university level, in Gage, N. L. Handbook on research on teaching. Chicago: Rand McNally and company.

McKeen, J. D. (1986). An experimental evaluation of alternate pedagogical approach for an introductory business data processing course. Computers & Education, 10(3), 369-374.

Nowaczyk, R. H. (1983). Cognitive skills needed in computer programming. (ERIC Document Reproduction Service No. ED 236 466).

Perkins, D. N., Hancock, C., Hobbs, R., Martin, F., & Simmons, R. (1986). Conditions of learning in novice programmers. Journal of Educational Computing Research, 2(4), 37-55

Sauter, V. L. 1986. Predicting computer programming skill. Computers & Education, 10(2), 299-302.

Schneiderman, B. (1980). Software psychology: Human factors in computer and information systems. New York: Winthrop.

Tsai, San-Yun, & Pohl, N. F. (1978). Student achievement in computer programming: Lecture vs. Computer-Aided Instruction. Journal of Experimental Education, 46(2): 66-70.

Walter, G. A., & Marks, S.E. (1981). Experiential learning and change, New York: John Wiley & Sons.

Wiggins, T. A. (1984). The effect of teaching method or student characteristics on student achievement or attitude in a BASIC computer programming undergraduate. Ph.D. Dissertation. Iowa State University. (University Microfilms International, Ann Arbor, Michigan).

Woodworth, R. S., & Scholosberg, H. (1954). Experimental psychology (rev. ed.). New York: Henry Hold and Company.

APPENDIXES

APPENDIX A

SUPLEMENTARY AID EXAMPLES (SAE)

OBJECTIVES:

Items to be learned are:

- o Start a new program
- o Display information at the terminal or on the screen.
- o Assigning information (Numerical and String) to Variables.
- o Skip or print blank lines.
- o End a program.
- o Save a program in a disk.

STATEMENTS:

NEW	*(Used to start a new program.)
CLS	(Used to CLear the Screen.)
REM or '	(Used for remarks. Everything following the REM will be ignored by the computer.)
PRINT	(Used to display information on the screen.)
+	(Used to add two numbers or two variables.)
=	(Used to assign data to variables.)
;	(Used to execute the statement that follows in the next available space.)
" "	(Used to hold a quote, a string of information to printed on screen or printer.)
END	(Used to terminate the program.)
LIST	*(Used to display the program in the screen.)
RUN	*(Used to execute the program.)
SAVE	*(Used to store the program in disk.)

PROGRAM:

```
OK
NEW
10 REM  Program to print values on the screen
20  CLS                                    'CLear Screen
30  A = 6
40  B = 5
50  C = A + B
60  PRINT " The value of A + B = ";C
70  PRINT                                  'Print a blank line in screen
80  PRINT A
90  PRINT B
100 PRINT A+B
110 PRINT C
120 END

LIST
SAVE  "ADD.BAS"
RUN
```

INSTRUCTIONS: Key program into memory. Run Program and make corrections. Change line 30 to A = 10 and line 40 to B = 15, then run program again.

* These statement used only to manipulate the program. Do not use in program.

OBJECTIVES:

Items to be learned are:

- o Entering information into the computer during the execution of the program.
- o The effect of semicolon ';' and commas ',' in the displacement of information on the line.
- o Relationship between entering and displaying information.
- o The manipulation of entered data.
- o List the program in the screen after it had been typed.
- o Obtain a listing of the program at the printer.

STATEMENTS:

INPUT	(Used to enter information from the keyboard)
*	(Used to multiply two numbers or variables)
LLIST	(Used to get a hard copy (a listing) of the program at the printer)

PRINT CLS NEW RUN SAVE "PROG1.BAS" LIST END

PROGRAM:

```
OK
NEW

10 CLS                                          'Clear screen.
20 PRINT "Hello !!!"
30 PRINT                                        ' A blank line will be printed.
40 INPUT "What is your first name ";NA$         ' The name will be stored in NA$
50 PRINT
60 PRINT "Nice to meet you ";NA$;"."
70 INPUT "How old are you? "; AGE               ' AGE is numeric value
80 ND = 365 * AGE                               ' 365 days X AGE = Age in days
90 CLS
100 PRINT
110 PRINT
120 PRINT "You are ";ND;" days old."
130 PRINT
140 PRINT "NICE TO MEET YOU ";NA$
150 PRINT
160 PRINT "Bye... Bye... Bye..."
170 END                                         ' End of the program.

LIST
RUN
SAVE "PROG1.BAS"
LLIST
```

INSTRUCTIONS: Key in this program, RUN it and and correct errors and analyse what is happening. SAVE program on disk. LLIST program on printer and turn in to instructor.

OBJECTIVES:

Items to be learned are:

- o Entering a single variable one at a time into the computer.
- o Manipulation of numeric data.
- o Using two or more variables in a calculation.
- o Displaying multiple variables in one line.
- o Loading any stored program from the disk to RAM.

STATEMENTS:

INPUT PRINT CLS END LIST SAVE RUN LLIST

PROGRAM:

```
OK
NEW

10 INPUT "Enter any number: ";N1                  ' Enter value of N1 from key board.
20 INPUT "Enter a second number: ";N2             ' Enter value of N2 from key board.
30 INPUT "Enter a third number: ";N3              ' Enter value of N3 from key board.
40 TL = N1 + N2 + N3                              ' TL = the total value.
50 PRINT "N1 = ";N1
60 PRINT "N2 = ";N2
70 PRINT "N3 = ";N3
80 PRINT
90 PRINT "The total of N1 + N2 + N3 = ";TL
100 PRINT
110 PRINT N1; " + " ; N2; " + " ;N3; " = " ;TL
120 PRINT
130 PRINT "END OF THE PROGRAM"
140 END

LIST
RUN
SAVE "NUMS.BAS"
LOAD "NUMS.BAS"
LIST
LLIST
```

INSTRUCTIONS: Key in this program and correct errors. RUN the program. Make changes by adding an INPUT statement to enter a fourth number and add this number to the total value TL. RUN the program again, SAVE it to disk, and then LLIST on printer. Give a hard copy to instructor.

OBJECTIVES:

Items to be learned are:

o Input several variables on the same line from the keyboard.

o Format line displacement.

o Clear the screen.

STATEMENTS:

INPUT X,Y,Z	(Input 3 variable values consecutively 'separated by commas')
PRINT X,Y,Z	(Used with commas to print variable values in different columns)
( )	(Used to calculate values with each other before other calculations)
/	(Used for divisions)

CLS LIST RUN SAVE

PROGRAM:

```
OK
NEW

10 CLS
20 PRINT "...THIS PROGRAM WILL CALCULATE THE AVERAGE OF THREE NUMBERS ..."
30 PRINT
40 INPUT "Enter X,Y,Z  separated by commas: ",X,Y,Z
50 AV = (X+Y+Z) / 3
60 PRINT
70 PRINT "The average of X, Y, Z = ";AV
80 END

LIST
RUN
SAVE "AVRGE.BAS"
LLIST
```

INSTRUCTIONS: Key in this program and correct errors. RUN the program. Make changes by adding any lines or statments to multiply AV (the average) by 2. RUN the program again and obtain a hard copy of the listing of the program and turn in to instructor.

OBJECTIVES:

Items to be learned are:

o Understanding loops.

o Branching form one line to another in both directions.

o Conditions (decisions making).

STATEMENTS:

IF THEN ELSE	(Used to check conditions)
TAB( )	(Used to space out a number of given columns)
STRING VARIABLES	(Used for any alphanumeric data. e.g. AN$, B$)
GOTO	(Used to branch from one line to another)
LPRINT	(Used to get a display at the printer)
LLIST	(Use to get a listing of program at the printer)
OR	(Its a logical OR, if one of the conditions is true, what follow THEN will be executed)

PRINT INPUT CLS SAVE

PROGRAM:

```
NEW

100 REM *** A practice in branching on conditions ***
110 CLS
120 PRINT
130 LPRINT
140 PRINT TAB(6);"STATES AND CAPITALS OF THE UNITED STATES"
150 LPRINT TAB(6);"STATES AND CAPITALS OF THE UNITED STATES"
160 PRINT  TAB(6);"----------------------------------------------------------------"
170 LPRINT TAB(6);"----------------------------------------------------------------"
180 PRINT
190 LPRINT
200 GOTO 220
210 PRINT "WRONG TRY AGAIN..":PRINT
220 INPUT "WHAT IS THE CAPITAL OF TENNESSEE";AN$
230 IF AN$="NASHVILLE" OR AN$="Nashville" THEN PRINT "CORRECT..." ELSE GOTO 210
240 PRINT
250 PRINT "You got it, the capital of Tennessee is Nashville."
260 LPRINT "You got it, the capital of Tennessee is Nashville."
270 END


LIST
RUN
SAVE "CAPITAL"
LLIST
```

INSTRUCTIONS: Change the state's and capital's names, and key in program. LLIST the program and turn in a hard copy to instructor.

OBJECTIVES:

Items to be learned are:

o Understanding loops.

o Branching form one line to another in both directions.

o Conditions (decisions making).

o Understanding loops within loops (nested loop).

STATEMENTS:

IF THEN ELSE

PRINT INPUT TAB( ) CLS : LLIST LIST RUN

PROGRAM:

```
NEW

100 REM *** A practice in branching and conditions ***
105 REM
110 CLS
120 PRINT
130 PRINT TAB(6);"STATES AND CAPITALS OF THE UNITED STATES"
140 PRINT TAB(6);"----------"
150 PRINT
160 GOTO 190
170 PRINT:INPUT "Wrong, would you like to try again";R$
180 IF R$="Y" OR R$="y" THEN GOTO 190 ELSE GOTO 210
190 PRINT:INPUT "WHAT IS THE CAPITAL OF TENNESSEE";AN$
200 IF AN$="NASHVILLE" OR AN$="Nashville" THEN PRINT "CORRECT..." ELSE GOTO 170
210 PRINT:PRINT "The capital of Tennessee is Nashville."
220 PRINT:PRINT:PRINT "        END OF THE PROGRAM"
230 END

RUN
SAVE "CAPITAL2.BAS"
LIST
LLIST
```

INSTRUCTIONS: Change the name of the state and then capital's name. Key in the program with the new modifications. LLIST the program and turn a hard copy to instructor.

OBJECTIVES:

Items to be learned:

o How to position the cursor in the screen.
o Using multiple commands or statements per line.
o Using ? symbol for PRINT.

STATEMENTS:

LOCATE X,Y	(Used to position the curser in the screen by row (1 to 24) and columns (1 to 80)).
:	(Used as a delimiter or a separator for each statement)
?	(Used to replace PRINT statement, ? will do the same thing as PRINT).

INPUT END

PROGRAM:

```
CLS
NEW

10 CLS : LOCATE 9,10 : INPUT "Please enter your name ";N$
20 LOCATE 15,10 : ? "Hi, nice to meet you";N$;"!"
30 END

LIST
RUN
SAVE "SCREEN.BAS"
```

INSTRUCTIONS: Key in this program and change the location of where information is to be printed, at line 10 and 20. Correct errors if any and RUN the program. LLIST the program and turn in a hard copy to instructor.

OBJECTIVES:

Items to be learned:

o Placing data (numeric or string) within the program for use at various stages throughout its execution. (Data Elements)
o Separating data elements with commas.
o Reading data elements (numeric or string) into variables and move the data marker or pointer to the next data element.

STATEMENTS:

DATA	(Used to allow information to be stored in a program for use at the execution of the program)
READ	(Used to assign data elements to variables and move the pointer)
,	**(The comma here is used to separate data elements or values)

NEW CLS PRINT LOCATE END LIST LLIST RUN SAVE LOAD

PROGRAM:

```
NEW

10 CLS
20 DATA Lori,Mike,Steve                    ' Can be placed at any line in the program.
30 READ N1$ : LOCATE 5,5 : PRINT N1$
40 READ N2$ : LOCATE 7,5 : PRINT N2$
50 READ N3$ : LOCATE 9,5 : PRINT N3$
60 LOCATE 11,5 : PRINT N1$;", ";N2$;", And ";N3$;" are my friends."
70 END

LIST
RUN
SAVE "PROG9.BAS"
NEW                              ' No lines will be listed.
LOAD " PROG9.BAS"          ' Bring the program from the disk.
LIST
LLIST
```

INSTRUCTIONS: Key in this program and change data elements in the DATA statement at line 20. RUN the program and LLIST it. Turn in a hard copy to instructor.

** Never put a comma at the end of DATA statement

OBJECTIVES:

Items to be learned:

- o Rereading the same data in a DATA statement more than one time
- o Reading multiple values (data elements) with a single read statement

STATEMENTS:

RESTORE (Used to move the data pointer to point at the first element in the first DATA statement)

DATA READ CLS PRINT IF THEN GOTO LOCATE

PROGRAM:

```
CLS
NEW

10 CLS:C=0                                              'C is a counter
20 DATA "Computers ", "are ", "fun to work with"
30 READ A1$,A2$,A3$
40 PRINT:PRINT:PRINT:PRINT A1$;A2$;A3$
50 C = C + 1
60 PRINT "C = ";C
70 IF C > 5 THEN 100
80 RESTORE
90 GOTO 30
100 LOCATE 22,10: PRINT " WAS'NT THAT FUN . . ."
110 END

LIST
RUN
SAVE "READDAT.BAS"
LLIST
```

INSTRUCTIONS: Key in this program and change data elements in the DATA statement. RUN the program and LLIST it. SAVE the program and turn in a hard copy to instructor.

OBJECTIVES:

Items to be learned:

- o Controlling the number of BASIC statements to be executed consecutively (number of loops).
- o Starting and ending the loops.
- o Increment or decrement of the loop counter.
- o Recalling a stored file.

STATEMENTS:

FOR	(Used to define the variables that is used as a counter that control the times of repetition)
NEXT	(Used with FOR statements to terminate or repeats the loops based on the upper or lower limits)
LOAD	(Used to load a stored program from any disk drive)

CLS INPUT PRINT

PROGRAM:

```
CLS
NEW

10 CLS
20 INPUT "ENTER YOUR NAME " NA$
30 FOR C = 1 TO 20            'this loop will repeat 20 times
40     LOCATE C,12+C
50     PRINT NA$
60 NEXT C
70 END

LIST
RUN
SAVE "SCR2.BAS"
LOAD "SCR2.BAS"               ' LOAD is only used if the file have been saved before
LIST
CLS
RUN
LLIST
```

INSTRUCTIONS: Key in this program and RUN it. Change the upper limits of the loop form 20 to 15, and RUN the program again. SAVE the program and LOAD it. LLIST the program and turn in a hard copy to instructor.

OBJECTIVES:

Items to be learned:

o Relational operators (=, <, >, <>) and their functions.

STATEMENTS:

=	(A = B)	(A is equal to B)
<	(A < B)	(A is less than B)
>	(A > B)	(A is greater than B)
<>	(A <> B)	(A is not equal to B)

PRINT INPUT CLS LIST RUN SAVE

PROGRAM:

```
NEW
CLS

10 REM   This program will compare the value of A  to the value of  B
20 REM    and print  thier relationship.
30 CLS
40 INPUT  " Enter a value for A: ",A
50 INPUT  " Enter a value for B: ",B
60 PRINT
70 IF  A>B  THEN PRINT " A IS GREATER THAN B."
80 IF  A<B  THEN PRINT " A IS LESS THAN B."
90 IF  A=B  THEN PRINT " A IS EQUAL TO B."
100 PRINT
110 IF  A<>B  THEN PRINT " A  AND  B  ARE NOT EQUAL."
120 END

LIST
RUN
SAVE "RELOPER.BAS"
LLIST
```

INSTRUCTIONS: Key in this program and RUN it. SAVE the program and LLIST it. Turn in a hard copy to instructor.

OBJECTIVES:

Items to be learn:

- o Writing an interactive CAI like program.
- o How to pause and keep information in the screen for a period of time.
- o forcing a specific value (numeric or string) to be entered.
- o Using WHILE WEND statements.

STATEMENTS:

WHILE WEND	(Its similar to FOR NEXT statements. WHILE WNED loops used to repeatedly execute a block of statements for as long as a given condition is true)
<>	(Used to evaluate of two values are NOT EQUAL)

PRINT CLS TAB( ) INPUT LOCATE X,Y FOR NEXT : END

PROGRAM:

```
CLS
NEW

20  REM ************************************************************
30  REM This is an interactive program that will demonstrate the use of WHILE WEND
40  REM  statements. First this program will ask for the user's name and put it in the
50  REM  variable NA$. Then it will query two different numbers, add them, and give
60  REM  the result in the screen. The program will ask the user to enter a C or Q to
70  REM  Continue entering different numbers and get the result or Quit the process.
80  REM ************************************************************
90  REM
100 CLS:PRINT :LOCATE 2,10:PRINT "LET US PLAY A GAME"
120 PRINT TAB(10);:INPUT "What is your name: ",NA$
130  CLS:LOCATE 10,10:PRINT "Nice to meet you ";NA$
140 FOR C = 1 TO 1500 : NEXT C              ' This is how you make the computer pause.
150 WHILE  AN$  <>  "Q"
160     CLS: PRINT TAB(5) " ENTER ANY TWO NUMBERS AND I'LL GIVE YOU THE TOTAL"
170     PRINT : INPUT " ENTER 1st NUMBER: ",N1
180     PRINT : INPUT " ENTER 2nd NUMBER: ",N2
190     PRINT:PRINT : PRINT " THE RESULT IS "; N1+N2
200     LOCATE 22,15: INPUT " Enter C to continue or Q to quit: ",AN$
210     IF AN$ <> "C" AND AN$ <> "Q" THEN PRINT "Please enter C or Q only":GOTO 180
220 WEND
230 PRINT:PRINT  "  END OF THE PROGRAM"
240 END

LIST
RUN
SAVE "WWEND.BAS"
LLIST
```

INSTRUCTIONS: Key in this program and modify it to do the follwing:

- o Longer pause at line 140
- o repeat the WHILE WIND loop (line 150-220) while AN$ is not equal to S
- o LLIST the program and turn in a hard copy to instructor.

OBJECTIVES:

Items to be learned:
o Systematic way of naming a large number of variables (an **array**).
o An array can be used to store integer, real, and string values.
o All the data in a **single array** should be of the same type.
o Each array must be given a unique name (array name), and each data item (array element) must be given a specific location within that array.
o Determining the **dimension** or the size of a single array.

STATEMENTS:

A(I)	(A is the array name, and I is the subscript that indicate the position of a data item within array A).
DIM A(15)	(Set a side additional space in an array with a size of 16 elements).

REM CLS FOR NEXT READ DATA PRINT SAVE LIST

PROGRAM:

```
NEW
CLS

100 REM This program will show the use of a single array.
110 REM Each data element will be read and placed in the array.
120 REM Each array element will be printed after it is placed in the array.
130 REM
140 CLS
150 DIM A(15)
160 FOR I = 1 TO 15
170    READ A(I)
180    PRINT A(I)
190 NEXT I
200 END
1000 DATA 101,102,103,104,105,106,107,108,109,110
1010 DATA 111,112,113,114,115
1020 REM *** 4/20/87 BY A.A. ***

RUN
LIST
SAVE "ARR1.BAS"
```

INSTRUCTIONS: Key in this program, and run it. Increment the array dimension to be 20 at line 150, change the upper limit of I to 20 at line 160, and add more data elements to be read at the DATA statment at line 1010. Run the program after all changes and turn in a hard copy of the program to the instructor.

OBJECTIVES:

Items to be learned:
o Systematic way of naming a large number of variables (an **array**).
o An array can be used to store integer, real, and string values.
o All the data in a single array should be of the same type.
o Each array must be given a unique name (array name), and each data item (array element) must be given a specific location within that array.
o Determining the **dimension** or the size of a single array.
o The use of **subroutines** (A subroutine is part of a program that can be executed any number of times in the same program.
o GOSUB and RETURN are always used together.

STATEMENTS:

X(I)	(X is the array name, and I is the subscript that indicate the position of a data item within array X).
DIM X(15)	(Set a side additional space in an array with a size of 16 elements).
GOSUB 1000	(Send the control of the program to line 1000).
RETURN	(Returns the control of the program to the statement directly under the GOSUB statement).

REM CLS FOR NEXT READ DATA PRINT RUN SAVE LIST END

PROGRAM:

```
NEW
100 REM This program will show the use of a single array.
110 REM Each array element will be entered from the key board.
120 REM All array element will be printed after they all have been placed in the array.
130 REM
140 CLS
150 DIM X(15)
155 REM *** Routine to input data from the screen.
160 FOR I = 1 TO 15
170    PRINT "ENTER VALUE #";I; : INPUT X(I)
180 NEXT I
190 GOSUB 1000 : CLS
200 REM *** Routine to print all the array elements.
210 FOR I = 1 TO 15
220    PRINT "#";I;"= ";X(I)
230 NEXT I
240 END
990 REM ** The following is a subroutine that forces the program to pause until you press RETURN**
1000 LOCATE 23,10 : INPUT "Press RETURN to continue";R$
1010 RETURN

RUN
SAVE "ARR2.BAS"
```

INSTRUCTIONS: Key in this program and make the following changes:
- o Display on the screen each array element after you enter it. (e.g. PRINT X(I)).
- o Make changes at line 210 so the computer will only print array elements 5 to 10
- o RUN the program again and turn in a hard copy to the instructor.

OBJECTIVES:

Items to be learned:

o Using more than one single array in the same program.

STATEMENTS:

A(I)	(A is the array name, and I is the subscript that indicate the position of a data item within array A. Only numeric data can be stored in A(I)).
N$(I)	(This is another type of variable, an array called N$. Data stored in N$(I) will be a string data.)

REM CLS DIM FOR NEXT READ DATA PRINT

PROGRAM:

```
NEW
CLS

100 REM This program will show the use of more than one single array.
110 REM Data elements will be read in pairs. A table will be printed
120 REM showing all the element in both arrays.
130 REM
140 CLS
150 DIM N$(10), A(10)
160 PRINT " I"," NAME","TEST1"
170 PRINT " -"," --------","---------"
180 FOR I = 1 TO 10
190    READ N$(I), A(I)
200    PRINT I,N$(I),A(I)
210 NEXT I
220 END
1000 DATA "Gregory",95,"Steve",70
1010 DATA "Lamar",90,"Jana",89
1020 DATA "Angela",86,"Mike",75
1030 DATA "Frances",97,"Carla",80
1040 DATA "Smith",86,"Linda",91
1050 REM *** 4/20/87 BY A.A. ***


RUN
LIST
SAVE "ARR3.BAS"
```

INSTRUCTIONS:

o Key in this program and correct errors if any.
o Add two more names and their scores to the data.
o Make correct changes at lines 150 and 180 so all the data will be read.
o Change PRINT statments to LPRINT and run the program.
o Turn in a hard copy of the program listing and the program output to instructor.

APPENDIX B

INFORMATION SHEET (DEMOGRAPHIC SURVEY)

INFORMATION SHEET

Please fill in the blanks and use "X" in between parenthesis.

SSN: _____ ____ _____

Major: _____.

Age: ( ) less than 25 ( ) 25 or older

Sex: ( ) male ( ) female

Present study classification:
( ) Undergraduate ( ) Special
( ) Graduate (Degree _____.)

Have you had any programming language course?
( ) none ( ) 3 courses
( ) 1 course ( ) more than three
( ) 2 courses

How much formal instruction have you had on using micro-computers?
( ) none ( ) 6 months - a year
( ) less than 6 months ( ) more than a year

How much formal instruction have you had on using mainframe computers?
( ) none ( ) 6 months - a year
( ) less than 6 months ( ) more than a year

Place a check for each mathematics course you have completed with a 'C' grade or better:

Prior to college	In college
( ) General math	( )( )( ) Math 0100-0120 (High school math)
( ) Algebra I	( )( )( ) Math 0130-0140 (Geometry)
( ) Geometry	( ) Math 0150 (Trigonometry)
( ) Algebra II	( ) Math 1140 (Basic math skills)
( ) Trigonometry	( )( )( ) Math 2110-2130 (Structure the number system)
( ) Analysis	( ) Math 1540 (College algebra)
( ) Pre calculus	( ) Math 1700 (Pre calculus)
( ) Advance math	( )( )( ) Math 1840-1860 (Calculus)
( ) Calculus	Other (Specify): _____.

What is/was your undergraduate GPA?
( ) below 2.00 ( ) 3.10 - 3.50
( ) 2.00 - 2.5 ( ) above 3.60
( ) 2.60 - 3.00

What factors most influenced you to attend this course?
( ) required ( ) to learn BASIC
( ) personal interest ( ) for experience
( ) influence by others
( ) other _____.

APPENDIX C

KNOWLEDGE TEST (PRETEST AND POSTTEST)

NAME_____

TRUE OR FALSE: Use a (+) for TRUE and a (0) for FALSE!

_____1. The Microsoft BASIC interpreter requires that each program statement be on a separate numbered line.

_____2. Variable values can be presented on the screen with the PRINT statement.

_____3. Once a diskette is FORMATTED, it can be used in any disk drive and be inserted in any direction.

_____4. The Disk Operating System keeps track of where programs are stored on a formatted diskette.

_____5. When a program line begins with IF, everything following on that line will execute only if the IF condition is true.

_____6. A line such as: 100 LPRINT "This is a message." will cause the quoted string to appear on the screen as well as the printer.

_____7. A loop caused by a GOTO statement always will execute faster than a loop caused by a FOR ... NEXT pair of statements.

_____8. The INPUT statement will cause the computer to pause and wait for an input from the keyboard, followed by the RETURN/ENTER key.

_____9. It is possible to cause the computer to pause for several seconds then resume operation without a keyboard response.

____10. The INPUT statement, followed by a quoted prompt line, is equivalent to a PRINT statement with a quoted prompt line followed by an INPUT statement.

____11. The following program line will cause a question mark to be inserted when the prompt line appears on the screen.

100 INPUT"What is the age of your brother";AGE

____12. The colon (:), when used on a program line, will cause whatever is to be printed next to be printed in the next available space on the line.

____13. Menu-driven programs generally allow the programmer to organize the program flow better and cause program exit only from the menu.

____14. The DATA statement is used only in conjunction with the READ statement.

____15. The RESTORE statement is used to clear memory of any variable values.

____16. Microsoft BASIC allows several program statements on the same numbered line as long as they are separated by colons.

____17. Proper use of the PRINT USING statement and formatting string will allow a number to be formatted to round off as well as to properly place a $.

____18. A GOSUB ... RETURN combination will cause a routine of statements to be executed and always return to the next statement after the GOSUB that caused it to happen.

____19. The quotation mark (") is ASCII code 34, however, one can cause a quotation mark to be printed on the screen by the statement:

120 PRINT (")

____20. A subroutine always is terminated with a RETURN statement.

____21. The program line: 200 A=2:PRINT A^3 will cause the number 6 to be printed on the screen.

____22. When one attempts to READ more DATA items than are present, an error message will appear that says "INPUT PAST END OF FILE."

____23. SAVEing a program on the diskette in ASCII format will produce a program file on the disk that is larger than the same program saved in the normal compressed binary format.

____24. The statement: RUN "B:HONORS.BAS" will cause the program in memory which you are going to call HONORS.BAS to run.

____25. File names in Microsoft BASIC can have up to eight characters and an extension of up to three characters (usually .BAS).

____26. To delete a program on drive B: called HONORS.BAS, one should issue the command KILL HONORS.BAS.

____27. When a printer is connected to the computer, but is not turned on, a program statement LPRINT will cause the program to halt operation.

____28. For a computer program in BASIC to return the control of the computer from BASIC back to the operating system, the last statement must be END.

____29. To List a program on the lineprinter, the statement issued must be LLIST, and the printer must be on, attached, and have the proper printer driver booted up from the disk operating system.

____30. When a program line contains: 100 DEFSTR A this means that the variable A is automatically a string, even without the $ designator.

____31. If a variable is dimensioned for 20 items, e.g. DIM B$(20), this means that there are going to be 20 items in memory all the time.

____32. A type mismatch error means that the program realizes it is being called upon to treat a string as a number or a number as a string.

____33. A program line states: 100 ON K GOSUB 1000,2000,3000,110
If the value of K is 2, the program execution would move to line 1000.

34. There are several mistakes in the program listed below. If the program line contains a mistake, circle it and correct the error or rewrite the line correctly.

```
10. REM This program is called SAMPLE.BAS
20  PRINT"Today is a day to practice troubleshooting programs."
30  INPUT"What name do you wish to use?":N$
40  PRINT"Hello there ";N$;" I have been waiting for you."
50  INPUT"Would you like to (1) Do arithmetic, or (2) Check spelling";AN
60  IF AN<1 OR AN>2 THEN PRINT"Your answer must be a (1) or (2).":GOTO 50
70  ON AN GOSUB 100,200
80  INPUT"Do you want to try another";T$:IF T="YES" THEN 50
90  END
100 INPUT"Type 2 numbers separated by commas.",N,R
110 PRINT"The product of";N;"and";R;"is";N*R
120 INPUT"Want another";Q$:IF Q$= YES THEN 110 ELSE RETURN
200 INPUT"Type the name of an animal that barks.".8
210 IF B$="DOG  THEN PRINT"You know your animals."
220 RESTORE
```

The following program is to be used to answer the questions beginning with number 35.

```
10 REM ** SAMPROG.BAS ** DEMONSTRATION PROGRAM FOR MIDTERM 5150 **
15 CLS
20 CLEAR
30 INPUT"What is your first name";N$
40 PRINT"Well ";N$;" we have an experience for you."
50 READ A$,B
60 PRINT A$;" has done us a favor.  He got";B"% on the first exam, and that"
70 READ A$,B
80 PRINT"helps our average.  You need to work on ";A$;", however, because she"
90 PRINT"is going to make you work for a grade with that";B;"%."
100 DATA GEORGE,60,ANN,90
110 GOSUB 200
120 PRINT:PRINT N$;", what is the sum of";C;"and";D;:INPUT AN
130 IF AN=C+D then Q=Q+1:GOTO 150
140 IF AN<>C+D THEN PRINT"That answer won't help the average any.":W=W+1
150 INPUT"Want another (Y or N)";ANS$:IF ANS$="Y" THEN GOSUB 200:GOTO 120
160 PRINT"You got";Q;"right and";W;"wrong."
170 END
200 INPUT"Will you please type two numbers separated by commas";C,D
210 RETURN
```

____35. Which line causes the subroutine at line 200 to be executed the first time?
____36. On which line are the values of ANN and 90 assigned to A$ and B?

____37. Which line has the statement that causes the computer to leave a blank line on the screen?
____38. Which line will be skipped when an IF condition is true?

____39. The number of wrong answers are summed up on which line?

____40. Which line contains the information needed in line 50?

____41. What is the value of B in line 70?

____42. What is the variable used to keep track of the number correct?

____43. Which line clears the memory of any variable values that might be stored?
____44. Which line completes the action of a subroutine?

____45. Which line causes the screen to be cleared?

____46. Which line describes the reason for the program?

____47. What line evaluates the contents of a string variable?

____48. On what line are two arithmetic operations performed?

____49. On which line are the values of the right & wrong accumulators printed?
____50. What is a string variable for which a name is assigned?

APPENDIX D

HUMAN SUBJECT ; APROVAL FORM

THE UNIVERSITY OF TENNESSEE
KNOXVILLE

Office of
Research
Compliances

March 16, 1987

Dr. John I. Mathews
415 Claxton Addition
CAMPUS

Mr. Ahmed S. Al-Ghamdi
UT P.O. Box 16309
CAMPUS

Dear Dr. Mathews and Mr. Al-Ghamdi:

The project which you submitted entitled, "Comparison of Two Teaching Methods for Learning the BASIC Language," CRP #2314-A, has been reviewed and certified exempt from review by the Committee on Research Participation.

This certification is for a period ending March 16, 1988. Please make timely submission of renewal or prompt notification of project termination (see item #2 below).

The responsibility of the project director includes the following:

1. Prior approval from the Director of Research Compliances must be obtained before any changes in the project are instituted.

2. A statement must be submitted (Form D) at 12-month intervals attesting to the current status of the project (protocol is still in effect, project is terminated, etc.).

The Committee wishes you success in your research endeavors.

Sincerely,

L. B. Cebik, Director

cc: Dr. Thomas C. Collins, Vice Provost for Research
Dr. Gerald D. Cheek

404 Andy Holt Tower / Knoxville, Tennessee 37996-0140 / (615) 974-7697

VITA

Ahmed Mohammed Sahab Al-Ghamdi was born in 1960 in Al-Baha, Saudi Arabia. He attended public school (elementary, intermediate, and secondary) in the same city. In July 1977, he was granted a scholarship from the Saudi government to continue his higher education studies in the United States of America. In December 1982, he received a Bachelor of Science degree in Electronics Technology from Indiana State University. He then earned his Master of Science degree in Industrial Professional Technology from the same university in May 1984.

He entered Graduate School at the University of Tennessee, Knoxville, in the Fall of 1984. He held a Graduate Assistantship as an information-center consultant and system analyst at the University of Tennessee Computing Center (UTCC), from October 1985 until December 1987. Most of his work at UTCC was concentrated on the VAX system and Data Base Management Systems.

In December 1987, he received his Doctor of Philosophy Degree majoring in Technological and Adult Education with a cognate in Computer Science from the University of Tennessee, Knoxville.