

1989

An investigation of the potential of guided Logo programming instruction for use in the development and transfer of analogical reasoning

Nealy Frank Grandgenett
Iowa State University

Follow this and additional works at: <https://lib.dr.iastate.edu/rtd>

 Part of the [Communication Technology and New Media Commons](#), [Curriculum and Instruction Commons](#), and the [Instructional Media Design Commons](#)

Recommended Citation

Grandgenett, Nealy Frank, "An investigation of the potential of guided Logo programming instruction for use in the development and transfer of analogical reasoning" (1989). *Retrospective Theses and Dissertations*. 9188.
<https://lib.dr.iastate.edu/rtd/9188>

This Dissertation is brought to you for free and open access by the Iowa State University Capstones, Theses and Dissertations at Iowa State University Digital Repository. It has been accepted for inclusion in Retrospective Theses and Dissertations by an authorized administrator of Iowa State University Digital Repository. For more information, please contact digirep@iastate.edu.

INFORMATION TO USERS

The most advanced technology has been used to photograph and reproduce this manuscript from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps. Each original is also photographed in one exposure and is included in reduced form at the back of the book. These are also available as one exposure on a standard 35mm slide or as a 17" x 23" black and white photographic print for an additional charge.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

U·M·I

University Microfilms International
A Bell & Howell Information Company
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA
313/761-4700 800/521-0600

Order Number 8920133

**An investigation of the potential of guided Logo programming
instruction for use in the development and transfer of analogical
reasoning**

Grandgenett, Nealy Frank, Ph.D.

Iowa State University, 1989

U·M·I
300 N. Zeeb Rd.
Ann Arbor, MI 48106

**An investigation of the potential of guided Logo programming
instruction for use in the development and transfer
of analogical reasoning**

by

Nealy Frank Grandgenett

**A Dissertation Submitted to the
Graduate Faculty in Partial Fulfillment of the
Requirements for the Degree of
DOCTOR OF PHILOSOPHY**

**Department: Professional Studies in Education
Major: Education (Curriculum
and Instructional Technology)**

Approved:

Signature was redacted for privacy.

In Charge of Major Work

Signature was redacted for privacy.

For the Major Department

Signature was redacted for privacy.

For the Graduate College

**Iowa State University
Ames, Iowa
1989**

TABLE OF CONTENTS

	PAGE
CHAPTER I: INTRODUCTION	1
Background of the Problem	2
Statement of the Problem	8
Goals of the Study	9
Research Questions	10
Limitations of the Study	10
Definition of Terms	11
Summary	14
CHAPTER II: LITERATURE REVIEW	17
Programming and the Development of Cognitive Skills	18
Guiding the Transfer of Cognitive Skills from Programming	27
Analogical Reasoning and Its Instruction	35
The Relationship of Analogical Reasoning to Programming	51
Summary	60
CHAPTER III: METHODS AND PROCEDURES	63
Subjects	64
Treatment Groups	68
Research Instruments	75
Research Design and Procedures	88
Directional Hypotheses and Analysis of Data	95
Summary	103

CHAPTER IV: RESULTS	107
Hypothesis One Results	108
Hypothesis Two Results	110
Results for the LogoWriter Basis Comprehension Test	113
Auxiliary Results	114
Summary of Study Results	127
TABLES FOR AUXILIARY RESULTS	131
CHAPTER V: DISCUSSION OF RESULTS	147
Summary of the Study	147
A Discussion of Far Transfer Results	150
A Discussion of Near Transfer Results	158
A Discussion of Basic LogoWriter Comprehension	164
Implications of Auxiliary Descriptive Statistics	166
Summary of Conclusions and Research Recommendations	169
Concluding Remarks	172
BIBLIOGRAPHY	174
ACKNOWLEDGEMENTS	186
APPENDIX A: SAMPLE BACKGROUND QUESTIONNAIRE	188
APPENDIX B: EXPERIMENTAL LARGE GROUP ACTIVITY SHEETS	190
APPENDIX C: CONTROL LARGE GROUP ACTIVITY SHEETS	196
APPENDIX D: EXPERIMENTAL LAB ACTIVITY SHEETS	202
APPENDIX E: CONTROL LAB ACTIVITY SHEETS	214

APPENDIX F: ANALOGICAL REASONING INSTRUCTIONAL TECHNIQUE	229
APPENDIX G: EXPERIMENTAL LARGE GROUP INSTRUCTOR OUTLINES	231
APPENDIX H: CONTROL LARGE GROUP INSTRUCTOR OUTLINES	243
APPENDIX I: EXPERIMENTAL LAB INSTRUCTOR OUTLINES	255
APPENDIX J: CONTROL LAB INSTRUCTOR OUTLINES	264
APPENDIX K: REUSE OF SUBPROCEDURES PROGRAMMING TEST	272
APPENDIX L: LOGOWRITER BASIC COMPREHENSION TEST	274
APPENDIX M: ANALOGICAL REASONING INTRODUCTION TRANSPARENCIES	287

LIST OF TABLES

	Page
TABLE 1: <u>Cognitive Ability Test Nonverbal Battery</u> KR-20 Reliability Estimates Computed for the Current Study	87
TABLE 2: Hypothesis 1 <u>Cognitive Ability Test - Nonverbal Battery</u> Comparison of Mean Composite Score for Experimental and Control Treatment Groups	110
TABLE 3: Hypothesis 2 <u>Reuse of Subprocedures Programming Test</u> Descriptive Statistics for the Reuse of Subprocedures Raw Score for Both Treatment Groups	111
TABLE 4: Hypothesis 2 <u>Reuse of Subprocedures Programming Test</u> Comparison of Means for the Transformed Reuse of Subprocedures Scores for Both Treatment Groups	112
TABLE 5: Instructional Content Comprehension <u>LogoWriter Basic Comprehension Test</u> Comparison of Means Scores for Both Experimental and Control Treatment Groups	114
TABLE 6: Auxiliary Results Hypothesis 1 <u>Cognitive Ability Test - Nonverbal Battery</u> Comparison of Mean Composite Score for Both Treatment Groups with Age and Computer Nervousness Controlled as Covariates	132
TABLE 7: Auxiliary Results Hypothesis 1 <u>Cognitive Ability Test - Nonverbal Battery</u> Comparison of Mean Composite Score for Both Treatment Groups by Gender, with Age and Computer Nervousness Controlled as Covariates	133

TABLE 8:	Auxiliary Results Hypothesis 1 <u>Cognitive Ability Test - Nonverbal Battery</u> Comparison of Mean Composite Score for Both Treatment Groups by Year in College, with Computer Nervousness Controlled as a Covariate	134
TABLE 9:	Auxiliary Results Hypothesis 1 <u>Cognitive Ability Test - Nonverbal Battery</u> Comparison of Mean Composite Score for Freshmen in Both Treatment Groups with Age and Computer Nervousness Controlled as Covariates	135
TABLE 10:	Auxiliary Results Hypothesis 1 <u>Cognitive Ability Test - Nonverbal Battery</u> Comparison of Mean Composite Score for Sophomores in Both Treatment Groups with Age and Computer Nervousness Controlled as Covariates	136
TABLE 11:	Auxiliary Results Hypothesis 1 <u>Cognitive Ability Test - Nonverbal Battery</u> Comparison of Mean Composite Score for Juniors in Both Treatment Groups with Age and Computer Nervousness Controlled as Covariates	137
TABLE 12:	Auxiliary Results Hypothesis 1 <u>Cognitive Ability Test - Nonverbal Battery</u> Comparison of Mean Composite Score for Seniors in Both Treatment Groups with Age and Computer Nervousness Controlled as Covariates	138
TABLE 13:	Auxiliary Results Hypothesis 2 <u>Reuse of Subprocedures Programming Test</u> Comparison of Means for the Transformed Reuse of Subprocedures Scores for Both Treatment Groups, with Age and Computer Nervousness Entered as Covariates	139
TABLE 14:	Auxiliary Results Hypothesis 2 <u>Reuse of Subprocedures Programming Test</u> Comparison of Means for the Transformed Reuse of Subprocedures Scores for Both Treatment Groups by Gender, with Age and Computer Nervousness Entered as Covariates	140

TABLE 15: Auxiliary Results Hypothesis 2 <u>Reuse of Subprocedures Programming Test</u> Comparison of Means for the Transformed Reuse of Subprocedures Scores for Males in Both Treatment Groups, with Age and Computer Nervousness Entered as Covariates	141
TABLE 16: Auxiliary Results Hypothesis 2 <u>Reuse of Subprocedures Programming Test</u> Comparison of Means for the Transformed Reuse of Subprocedures Scores for Both Treatment Groups by College Year, with Computer Nervousness Entered as a Covariate	142
TABLE 17: Auxiliary Results <u>Programming Instrument Descriptive Statistics</u> Percent of Treatment Group Getting Specific Problems Correct on the Reuse of Subprocedures Programming Instrument	143
TABLE 18: Auxiliary Results <u>Programming Instrument Descriptive Statistics</u> Mean Number of Commands Used Per Successful Program on the Reuse of Subprocedures Programming Instrument	144
TABLE 19: Auxiliary Results <u>Programming Instrument Descriptive Statistics</u> Percent of Treatment Group Using Variables and Recursion Within the Reuse of Subprocedures Programming Instrument	145
TABLE 20: Auxiliary Results <u>Correlations Between Outcome Variables</u> Correlations of the Cognitive Ability Test - Nonverbal Battery With Selected Programming Variables on the Reuse of Subprocedures Programming Instrument	146

LIST OF FIGURES

	Page
FIGURE 1: Sequence of study events	90
FIGURE 2: Interaction of college year with instructional treatment	153

CHAPTER I: INTRODUCTION

In 1933 John Dewey suggested that the major purpose of education was teaching people to think. He stated that "education...is vitally concerned with cultivating the attitude of reflective thinking, preserving it where it already exists, and changing looser methods of thought into stricter ones whenever possible" (p. 78). As we approach the 21st century, such a purpose for education may be all the more relevant. We are entering what has been coined the "Information Age", where information is replacing land, labor, and capital as the most important societal commodity (Stonier, 1983). Human knowledge is expanding at an incredible rate, and the efficient management of that information is becoming of primary importance to our society and its educational institutions. The ability to think in a careful and logical manner, as suggested by Dewey, would seem to be increasingly important in this new age.

In a book edited by Mary Alice White, What Curriculum for the Information Age?, author Julie McGee suggested that there was a fundamental need to "reorder the curriculum to emphasize a new hierarchy of skills; skills that will equip students for life in the Information Age" (1987, p. 82). She stated that the central skills needed are: "the ability to evaluate information, the ability to set priorities, and the ability to make decisions (p. 82). Therefore, critical thinking, and its associated problem solving and informational skills,

would seem to be an emerging focus for education in this new era. Such a focus implies that the careful and rapid investigation of potential methods to effectively instruct these cognitive skills will be of utmost importance to the changing curriculum of this new age.

Background of the Problem

Critical thinking involves a variety of important cognitive skills. Although the specific skills comprising "critical thinking" is still in extensive debate (Halpern, 1987), one skill in particular, analogical reasoning, has been identified by a variety of researchers, such as Sternberg (1977b), Gick and Holyoak (1980), and Halpern (1987), as one of these specific skills. Analogical reasoning is fundamentally the ability to utilize a well understood problem to provide insight and structure for a less understood problem (Gentner, 1982). For example, when a student is learning about the structure of an atom, he or she might assist understanding by referencing previous learning about the structure of the solar system. Such reasoning would seem to permeate everyday life, as previous experiences are used to understand current situations, and former problems are referenced to gain insight into new ones.

In research investigating problem solving, analogical reasoning has been targeted as one of the most important problem solving processes that humans use (Hunt, 1982). Polya, in his work on mathematical

problem solving, and in his discussion on student conclusions, stated that "inference by analogy appears to be the most common kind of conclusion and it is possibly the most important" (1957, p. 43). Some researchers have even gone so far as to indicate that all problem solving can be seen as fundamentally analogical in nature; as learners continually attempt to transfer knowledge from a known situation to a novel one (Moore and Newell, 1973; Rumelhart and Norman, 1981).

Can students be taught to be good analogical reasoners in light of the global nature of this skill? This question has been posed by researchers, but seems to have received little serious research and empirical investigation (Holyoak, 1984). Most of the research which does exist has investigated analogical reasoning training within a controlled laboratory setting (e.g., Sternberg, Ketron, & Powell, 1982). Very few studies have investigated analogical reasoning training given within the dynamic environment of the classroom. Although theoretical support for the potential success of classroom training in analogical reasoning exists (Holyoak, 1984, Sternberg, 1977b), studies that actually attempt training in the classroom are greatly needed (Alexander, White, Haensly, & Crimmins-Jeanes, 1987).

General analogical reasoning training is not easy to incorporate into the classroom. In considering the use of analogy in problem solving, Holyoak suggests that analogical thinking probably can be improved by training, but that such training must incorporate a careful assessment of the way in which completed problems will be encoded

by the student. The important characteristic of a problem, as it relates to other similar or more general problems, may need to be emphasized in order for the student to be able to reference it in the solving of additional problems. Thus, students may need to be taught to explicitly note abstract goals, plans, and causal relations between problems they encounter to achieve skills which are not strictly domain specific (1983). Such teaching demands that a great deal of careful planning go into the instruction.

Recently, a direct attempt at instruction of analogical reasoning in the language arts classroom was made by researchers Alexander, White, Haensly, and Crimmins-Jeanes using 4th, 8th and 10th graders (1987). By using the Sternberg componential model (Sternberg, 1977a), analogical reasoning training was incorporated into an existing language arts curriculum. Students were gradually moved from concrete nonverbal analogies to more abstract verbal analogies by use of instruction based on Sternberg's component processes. Using the Woodcock Reading Mastery test as an outcome measure, these researchers found a significant effect for their classroom analogical reasoning training. Alexander, White, Haensly, and Crimmins-Jeanes felt their study demonstrated that analogical reasoning training could be provided in the uncertain and dynamic environment of the classroom. However, they were also careful to suggest that further research, especially within differing age groups and content areas, was

greatly needed to determine the effects of classroom-based componential training in analogical reasoning.

One research area which seems to offer potential for student development in generalized analogical reasoning is the continuing investigation of the cognitive demands and benefits associated with computer programming. Computer programming has long been investigated as a rich environment for the exercise of general critical thinking skills by researchers such as Papert (1980), and Feurzig, Horowitz, and Nickerson (1981). The Logo programming language has operated as a research focus in many of these investigations, and some positive cognitive benefits from the programming environment have been demonstrated in Logo studies (Howe, O'Shea, & Plane, 1979; Clements & Gullo, 1984; Degelman, Free, Scarlato, Blackburn, and Golden, 1986). However, the large majority of studies in this area have been anecdotal and case study in nature. There is a critical need for empirical classroom studies investigating the potential cognitive benefits of computer programming (Pea, 1983; Khayrallah & Maud Van Den Meiraker, 1987; Walker, 1987).

Within the last few years, the skill of analogical reasoning has begun to be specifically targeted in investigations involving computer programming benefits (Mann, 1986; Clement, Kurland, Mawby, & Pea, 1986, Swan & Black, 1987). Researchers Clement, Kurland, Mawby and Pea, at Bank Street College, recently found a positive correlation between success on an analogical reasoning task and the reuse of

subprocedures within a specific programming task (1986). The appropriate reuse of subprocedures, and the careful planning for a reuse of subprocedures, is essential for effective modular programming. Upon further investigation and discussion, these researchers concluded that "analogical reasoning may itself develop through its practice in programming" (p. 482).

Reasoning by analogy would seem to be an essential tool in the programming process. In considering the actual process of program construction, expert programmers seem to make strong use of their analogical reasoning ability to utilize past programming experiences in constructing an effective solution to a new problem (Pennington, 1982). These experts appear to reference quickly from internal "storehouses" of past programs, choosing or modifying relevant structures and plans, to facilitate the development of new programs. This behavior exhibited by these experts suggests that analogical reasoning may be a skill which is fundamentally tied to the general problem solving processes used in effective computer programming.

There is some indication however, that general problem solving and critical thinking skills that might develop through programming, including the skill of analogical reasoning, may need to be taught directly rather than be expected to occur spontaneously (Pea, 1983). Simply being involved in computer programming activities may not provide students with enough support to encourage the use of these general cognitive skills outside of the programming environment (Pea

& Kurland, 1987; Salomon & Perkins, 1987). Several studies that have looked specifically at the spontaneous transfer of cognitive skills from programming have generally failed to find significant effects on tasks outside of the programming domain (Pea, 1983; Pea & Kurland, 1984b). Apparently, if programming is to be used for developing generalized critical thinking skills, specific instruction or guidance in those skills may need to be incorporated into the programming instruction. Thus, if analogical reasoning developed through programming is expected to help a student solve problems in other non-programming domains, some special methodology focusing on the general nature of analogical reasoning may be needed in the programming instruction.

In an extensive recent investigation and review of the research literature regarding the potential cognitive outcomes to programming in Logo, Swan and Black (1987) found that virtually all studies reporting positive transfer results shared elements of a pedagogy specifically encouraging transfer. They suggested three common pedagogical transfer elements: 1) a focus on specific aspects of the problem solving process, 2) a direct instruction of the targeted skills, and 3) a mediated learning approach to student and teacher interaction. They indicated that these three pedagogical elements were typically present in studies where positive transfer into non-programming domains had been demonstrated. Thus, for computer programming to successfully develop a cognitive skill useful in another

domain, programming instruction may need to incorporate pedagogical elements that specifically encourage transfer to the new domain.

As teachers search for methods to prepare their students for the critical thinking needed in the information age, they will be focusing on methods that are useful in a classroom setting. Computer programming seems to offer the potential for efficiently improving the general skill of analogical reasoning in a class group; however, it appears that teachers will not be able to simply teach their students to write programs. Some careful attention to an instructional methodology supporting the general nature of analogical reasoning, and encouragement of the transfer of the skill to other domains, may be needed for this potential to be realized.

Statement of the Problem

Research about analogical reasoning is moving beyond theoretical definition and is beginning to search for methods to instruct and develop this important skill in learners. This is occurring at the same time that research about the cognitive benefits of computer programming is moving from the investigation of cognitive skills achievable spontaneously through programming, to a more deliberate focus on guiding the development of specific cognitive skills while programming. It may be that computer programming can act as a useful and powerful instructional medium for the development of

general analogical reasoning, but this potential is currently only a theoretical extension of the ongoing research from these two traditionally separate areas. Empirical investigation of this potential is needed. Thus, the problem of this study was to investigate the potential for the development of general analogical reasoning offered by the guided instruction of computer programming.

Goals of the Study

The two goals of the study were to look at two major potential effects of incorporating systematic analogical reasoning training within guided Logo programming instruction. The first goal was to investigate the far transfer effects of such programming instruction on general analogical reasoning development. The second goal of the study was to investigate the near transfer effects of such programming instruction on a related and important computer programming skill - the ability of the student to reuse subprocedures between programming problems.

Research Questions

There were two research questions for this study:

- 1) Would students experiencing Logo programming instruction, systematically oriented for the development of general analogical

reasoning, demonstrate greater analogical reasoning development than students experiencing Logo programming instruction not systematically oriented for transfer?

2) Would students experiencing Logo programming instruction, systematically oriented for the development of general analogical reasoning, demonstrate a greater tendency to reuse subprocedures between programming problems than students experiencing Logo programming instruction not systematically oriented for transfer?

Limitations of the Study

The study was conducted with acknowledgement to the following limitations:

1) It was necessary to operationalize the definition of analogical reasoning ability as the ability to solve problems in an "analogy type" format. It is recognized that analogical reasoning is a skill that extends beyond a narrow problem solving definition.

2) Due to a lack of suitable standardized instruments, it was necessary to use an investigator modified instrument, developed by previous researchers, for assessing the student reuse of subprocedures between programming problems.

3) Programming instruction was limited to the Logo programming language, thus generalizations to other programming languages and environments are restricted.

4) The sample was college students enrolled in an educational computing class, thus generalization to other populations is limited.

Definition of Terms

Logo programming - Logo is a computer programming language designed by Seymour Papert and associates at the Massachusetts Institute of Technology for use in a programming environment with educational applications. It is a list-processing language that combines formal procedural representations with concrete and immediate feedback to provide the student with an environment designed to facilitate intellectual exploration and experimentation. More than any other programming language, it has been the focus of a discussion by researchers on the cognitive benefits of learning and engaging in computer programming activities (Khayrallah & Van Den Meiraker, 1987). For this study, the software package LogoWriter, by the LCSi company, was used for Logo programming activities. LogoWriter has the same Logo structure available in other versions of Logo, but includes an improved editor, and expanded shape and turtle graphic capabilities.

Guided Logo programming instruction - Guided Logo programming instruction is defined as programming instruction, using the Logo programming language, that is systematically oriented for transfer of a specific cognitive skill. Based on the Logo research analysis of Swan

and Black (1987), this orientation involves an emphasis on three pedagogical components to facilitate transfer. These components are: 1) a focus on a specific cognitive skill, 2) direct instruction of that skill through programming, and 3) a mediational style of teaching by the instructor.

Direct instruction - Direct instruction is defined by Doyle (1983). Direct instruction is instruction which carefully structures cognitive tasks, utilizes explicit instruction of the problem-solving processes involved in those tasks, and incorporates a systematic guidance through a series of exercises that permit frequent opportunities for practice and feedback.

Mediational style of teaching - This study utilizes the definition of Delclos, Littlefield, and Bransford (1984) for a mediational style or approach to the teaching of Logo. In such a style, "the teacher makes specific and conscious attempts to frame what is learned in the Logo lesson in a broader context and to bridge specific principles learned to other situations where the same type of strategy would apply" (p. 6). Thus, in this approach the teacher continually seeks to help students formulate general principles from class activities, rather than principles specific to the immediate content. Also, in this approach students are helped to view themselves as active problem-solvers and learners, by being prompted to continually analyze their own thinking strategies, and by being encouraged to generate increasingly efficient

and alternative solutions to problems. This teaching approach typically utilizes Socratic dialogue between teacher and students.

Analogical reasoning - As suggested by Sternberg (1977a), people reason analogically whenever they "make a decision about something new in experience by drawing a parallel to something old." In this study, this important skill is operationally defined to encompass the ability to solve problems of an analogy format as found in the Cognitive Abilities Test - Nonverbal Battery (Thorndike & Hagen, 1986). This test utilizes geometric figures in three specific subtest formats and has been extensively normed and standardized. Analogy items, similar in format to this test, have long been included on many standardized, psychometric tests. The reliability of standardized analogy tests are typically high, and the indication of general analogical reasoning skill by tests using this format are reported as strongly valid. As stated by Sternberg in his research on the component processes of analogical reasoning, and the operation of these components in standard analogy tests: "The construct validity of performance on tests of analogical reasoning is unimpeachable" (1982, p. 237).

Transfer - As defined by Cormier and Hagman (1987), transfer of learning occurs "whenever prior-learned knowledges and skills affect the way in which new knowledges and skills are learned and performed" (p. 1). Transfer of learning is often divided into two types, near transfer and far transfer. Near transfer is usually considered to be transfer of learning which takes place within the same subject domain.

In this study, near transfer of learning was expected to occur if programming instruction, systematically incorporating analogical reasoning training, facilitated the reuse of subprocedures within the programming activity. Far transfer is usually considered transfer of learning which takes place between different subject domains. In this study, far transfer of learning was expected to occur if the Logo programming instruction, systematically incorporating analogical reasoning training, positively influenced scores on the Nonverbal Battery of the Cognitive Abilities Test.

Summary

This chapter introduced a study that investigated the potential of guided Logo programming instruction targeting the development and transfer of analogical reasoning. The chapter included a brief discussion on the background of the study, a statement of the study problem, goals for the study, research questions, limitations, and study definitions.

The background of the study was discussed with initial reference to the educational concerns of our society; a society now entering the information age. This societal age will require effective methods for instructing critical thinking and problem solving skills, and this study targeted analogical reasoning as one such skill. Classroom instruction of computer programming has been suggested by researchers as

offering potential for the development of analogical reasoning; however, researchers also imply that such potential may only be realized if the instruction is systematically guided to develop this skill.

Thus, the problem of this study was to investigate the potential for the development of general analogical reasoning offered by the guided instruction of computer programming. This potential is implied by: 1) research on analogical reasoning, and 2) research on the cognitive benefits of computer programming.

This chapter presented the two goals for the study: 1) the investigation of the far transfer effects of guided Logo programming on general analogical reasoning skill, and 2) the investigation of the near transfer effects of such instruction on student reuse of subprocedures between programming problems. These goals were related to two research questions. The first question asked whether students experiencing guided Logo programming instruction would demonstrate greater general analogical reasoning than students experiencing more traditional Logo instruction. Similarly, the second question asked whether these students experiencing guided instruction would also demonstrate a greater tendency to reuse subprocedures between programming problems than the more traditionally instructed group.

Finally, this study investigated the potential of guided Logo programming instruction for use in the development of analogical reasoning as a single step in helping to find possible methods to instruct general cognitive skills. It focused on analogical reasoning as

one specific skill, and guided Logo programming as one particular method. Research related to the potential development of analogical reasoning through guided Logo programming is discussed in Chapter Two.

CHAPTER II: LITERATURE REVIEW

The purpose of this chapter is to review previous research related to Logo programming instruction and the potential development and transfer of analogical reasoning. Various pedagogical implications for Logo instruction when seeking the transfer of cognitive skills to other domains beside programming will be discussed, with the potential for the development of analogical reasoning as a central focus to this investigation.

This review will be structured by focusing on four areas:

1) research concerning programming and the development of cognitive skills, 2) research on guiding the transfer of cognitive skills from programming, 3) research on analogical reasoning and its instruction, and 4) research regarding the relationship of analogical reasoning to programming.

Although this discussion will target the Logo programming language and the specific cognitive skill of analogical reasoning, more general research involving other programming languages, and other cognitive skills, will be incorporated when this research provides insight into the potential development of analogical reasoning through guided Logo programming.

Programming and the Development of Cognitive Skills.

Computer programming has often been a topic of discussion in educational computing research. As well as being seen as a potentially useful skill in future careers and occupations, it has been regarded by many researchers as a rich environment for the practice and development of general cognitive skills. A wide variety of such skills has been mentioned by researchers as possible cognitive benefits to active participation by a student in computer programming. These have included skills such as metacognition, general problem solving, procedural reasoning, divergent thinking, and general heuristics (Papert, 1972, 1980; Feurzig et al., 1981). It is this potential for cognitive development that seems to be a major reason for the continued popularity of computer programming in the general school curriculum, especially for the pre-high school grades where the occupational advantages of computer programming are less immediate (Pea & Kurland, 1984a).

Claims for the cognitive benefits of programming

Many of the claims for the potential cognitive benefits of computer programming have centered on the Logo programming language. Logo was developed by Seymour Papert and colleagues in the late 1960s at the Massachusetts Institute of Technology, as a computer programming language developed specifically for education. Since

that time, Logo has clearly become the language of focus in the discussion of programming's potential cognitive benefits (Khayrallah & Van Den Meiraker, 1987).

Logo, as seen by its creators, was more than a programming language. It was a carefully developed computer culture, where students could engage in self-guided Piagetian type learning. In this environment it is believed that students practice and develop important thinking skills in a natural non-threatening setting. Students theoretically use the programming language as an explicit medium for their own thinking, employing the computer as an "object to think with". They direct and teach the computer. This approach is significantly different from the more traditional computer-assisted instruction environments, where the student is typically the one taught or directed (Papert, 1980).

Specific claims concerning the potential cognitive benefits from programming, many referencing Logo, have not been modest. Feurzig, Horowitz and Nickerson (1981) have listed a large number of potential benefits including: more rigorous thinking, better understanding of general programming concepts, greater facility with heuristics, improved abilities with problem decomposition, and an enhanced self-consciousness with solving problems. Watt (1982) targeting the Logo language in particular, stated:

"[The Logo] programming environment can help children to develop problem solving skills, to think more clearly, to develop an awareness of themselves as thinkers and learners" (p. 48).

Often, such claims for the cognitive benefits of programming focus on the potential for problem solving development offered by computer programming as an explicit language for the problem solving experience. Thus, the programming language itself, which provides the means for students to communicate with the computer, can also provide students with the means to analyze and refine their own thinking about a specific problem. As expressed by Papert, such a situation can theoretically provide assistance to a student's cognitive operations:

"The child programs the computer. And in teaching the computer how to think, children embark on an exploration about how they themselves think. The experience can be heady: Thinking about thinking turns the child into an epistemologist, an experience not even shared by most adults" (1980, p. 19).

Initial investigations testing the claims of cognitive benefits

There has, however, along with the claims for the cognitive potentials of computer programming, been a lack of classroom studies to refine and test these claims. A few early researchers have attempted this. A study by Howe, O'Shea, and Plane (1979) found that a small group of 11 year-old students involved in Logo programming for a year did demonstrate slightly better understanding of certain algebra concepts than did a control group not involved in programming. Also, teachers in this study felt the Logo group could articulate mathematical issues and difficulties more efficiently. Unfortunately, this study was plagued by concerns related to a small sample size, and the possible bias introduced when teachers rated students while having knowledge of their experimental treatment.

The Brookline Logo project (Papert, Watt, diSessa, & Weir), also of 1979, is often cited as a early major project for the investigation of learning in the Logo environment. This study, funded by the National Science Foundation and conducted in the Brookline Massachusetts Public Schools, sought to document the behavior of sixty-six fourth to sixth graders while working with Logo and to correlate this work to geometric reasoning ability. The project directors found anecdotal evidence in the form of teacher testimonials for increased learning by the Logo group. However, empirical evidence, in the form of scores on the geometric reasoning tasks, was not statistically significant for the study.

An anecdotal basis for the overall beneficial effects for programming in Logo was also reported in an article reporting on the Lamplighter Project (Overall, 1981). This study listed a variety of cognitive benefits including problem solving skills, logical thinking skills, rule learning, and improved self-concept development. This article based these conclusions on non-experimental observations of the third grade participants of the study. Again, teacher testimonials, rather than empirical evidence, were the basis for the findings.

Empirical investigation of the cognitive benefits

Although much of the early research finding positive results for the cognitive benefits to computer programming relied on anecdotal evidence, a few empirical studies are available. Clements and Gullo, in a widely focused empirical study (1984), investigated the effects of learning Logo programming on various aspects of young children's cognition. The aspects examined included metacognitive ability, cognitive style, and cognitive development. The researchers used eighteen first grade students, randomly assigned to a Logo or CAI group, for a twelve week treatment. The treatment was rich in supervision, with a teacher present for every two or three students. Post-test results indicated a difference in favor of the Logo group in metacognition and divergent thinking. No differences were found for the groups on two Piagetian tasks of cognitive development.

In a more recent study (1986), Clements followed up this work with a study using six to eight year old children in a 22 week extensive study involving a Logo, CAI, and control group. Using a wide variety of empirical instruments, significant differences were found in favor of the programming group for operational competence, metacognition, creativity, and direction giving. No differences were found on measures of reading and mathematics achievement. Again, the Logo and CAI treatments were somewhat atypically rich in supervision with 3 pairs of students facilitated by one or two teachers during each session.

In a longitudinal look at the same children from the 1984 study (1987), Clements reported empirical results still favoring the Logo group on language mechanics and vocabulary, and in analogical reasoning. Thus, the Logo group still demonstrated a difference from the non-programming CAI group after an eighteen month period.

In an empirical dissertation study similar to the Clements studies, and utilizing careful sequencing of the programming tasks in the Logo instruction, Odom (1982) looked at groups of fifth and sixth grade volunteers given the Ross Test of Higher Cognitive Processes before and after instruction. Two groups were compared, one group experiencing Logo programming instruction and another group experiencing non-programming instruction. Significant differences favoring the Logo group were found for analysis and evaluation levels of

the Ross Test of Higher Cognitive Processes, with no differences found for the synthesis level.

Negative results for the spontaneous transfer of cognitive skills

Results from studies examining the potential cognitive benefits of computer programming have not always been positive however. In a discussion of the cognitive aspects programming, Pea and Kurland (1984a, 1987) have offered skepticism regarding the spontaneous transfer of cognitive skills from the computer programming environment. Generally, their skepticism regarding spontaneous transfer has developed from the fact that students often do not attain the degree of programming proficiency needed to support spontaneous transfer, and often draw little cognitive support for transfer in the discovery learning approach typical of Logo. In a study illustrating this problem, Pea (1983) looked carefully at the programming proficiency of fifty-nine to twelve year old children after a year of intensive programming. In this study children were assessed for program debugging, program writing, and program understanding. The results showed that students still had relatively poor skills in all areas even after a year of actual programming.

In a study targeting the potential development of general planning skills from Logo programming, Pea and Kurland (1984a) compared a Logo group to a non-programming group on a classroom planning task. The study used a group of nine to twelve year old children over the

period of a single school year. Study results showed no difference in the planning skills between the groups. In an immediate replication, with a different but similar group of children, Pea and Kurland (1984b) modified the planning task so that it was done on the computer to more directly simulate the Logo environment of immediate feedback. As in the first study, no differences were found between groups.

Pea and Kurland, although supportive of computer programming as an intellectual tool to practice cognitive skills, find fault with the discovery learning approach that has been typical of Logo (1984b, 1987). They clearly see an educational potential in programming, but see no guarantee offered for general cognitive development from the programming activity alone. It is this discovery learning approach of Logo, rather than the activity of programming itself, which has initiated much of the concern and criticism over the potential cognitive benefits to programming (Khayrallah & Van Den Meiraker, 1987).

A need for more empirical and narrowly focused research

Generally, research on the cognitive benefits of programming is an area filled with mixed and inconclusive results. Such a state is facilitated by the disproportionate ratio of a large number of anecdotal reports and testimonials to a relatively small number of empirical studies. The anecdotal studies generally indicate a potential for development of cognitive skills through programming, but make specific predictions or instructional implications difficult. It seems

apparent that this ratio needs to become more balanced, and indeed many researchers are indicating a critical need for empirical studies to help define and confirm the cognitive potential of programming (Pea & Kurland, 1984b; Khayrallah & Van Den Meiraker, 1987; Walker, 1987; Becker, 1987).

An additional concern in this research area deals with the wide focus of completed studies. Empirical studies, as well as anecdotal ones, have tended to utilize a more general investigative approach, looking at an assortment of cognitive skills and outcome measures, rather than at a specific problem solving or cognitive process. Such research is important for generating hypotheses regarding the cognitive benefits to programming, but provides relatively inconclusive results in determining the potential of programming in developing a specific cognitive skill. Also, many of the programming studies, both anecdotal and empirical, have often incorporated a relatively atypical classroom situation, with extremely high student teacher ratios, or fairly isolated small group instruction. Such research makes it difficult to derive much direction for the classroom teacher. As already indicated by a large number of researchers (Mann, 1986; Salomon & Perkins, 1987; Swan & Black, 1987), further systematic research focusing on specific cognitive skills, but incorporating a variety of different populations and pedagogies, is greatly needed.

Guiding the Transfer of Cognitive Skills from Programming:

Perhaps one of the reasons that research on the cognitive benefits of computer programming has been so widely focused and anecdotal in nature is the inherent complexity of the programming activity. The process of computer programming is indeed complicated, involving many difficult skills. These skills involve 1) understanding the task the program is to accomplish 2) planning a programming strategy to accomplish the task, 3) implementing a plan via a programming language, and 4) debugging a plan and the program code (Pea & Kurland, 1984a, 1987). Thus, computer programming requires a substantial amount of careful planning when developing a program to accomplish some task. Such planning behavior draws heavily on mental abilities, and it has been shown that expert programmers often use a large variety of cognitive skills when they participate in the retrieving and assembling of cognitive plans for a new program (Nichols, 1981).

The necessity of guiding planning skills from programming

Pea and his associates at Bank Street College found that students often avoid planning behavior completely when working on a computer program. They seemed to opt for a trial and error programming style that negated any need for higher cognitive processes (Pea, Kurland, & Hawkins, 1987). Therefore it is unrealistic to expect that students will

spontaneously develop generalized cognitive skills, when they seem to actively avoid the use of such skills when engaged in a programming task. For this reason, Pea has indicated that the problem solving and critical thinking skills attainable through computer programming may need to be taught directly rather than expected to occur spontaneously (Pea & Kurland, 1984a). Programming instruction may have to explicitly emphasize the targeted skills so that students are not permitted to rely solely on a preferred trial and error approach to writing programs.

Planning skills in particular seem unachievable in the programming environment spontaneously (Clements & Merriman, 1987, Pea & Kurland, 1984b). Children and novice programmers often slip into a "hacking" method of programming, during which they blindly try command after command, incorporating little reflective thought (Leron, 1985). Such a strategy may eventually achieve the desired output without the student ever incorporating anything but superficial planning into the problem solving process. Students may require instruction that directly encourages the planning process. As stated by Clements and Merriman: "to develop planning skills, it may be necessary to structure children's work so that they predict and plan before programming" (1987, p. 28).

The necessity of guiding other cognitive skills

Planning skills are not the only cognitive skills that may need instructional support in the programming process to encourage generalized transfer to non-programming tasks. Swan and Black (1987), extensively reviewed the literature regarding the cognitive outcomes to programming in Logo. They found that all studies noting positive results of cognitive skill transfer shared specific elements of a pedagogy that encouraged such transfer. They pointed to three pedagogical transfer elements: 1) a focus on specific aspects of the problem solving process, 2) direct instruction of the targeted skill, and 3) a mediated approach to student and teacher interaction.

Swan and Black (1987) attempted to design instruction around these pedagogical components in a twelve week study involving six successive units dealing with six cognitive skills: 1) subgoal formation, 2) forward chaining, 3) backward chaining, 4) systematic trial and error, 5) alternative representation, and 6) analogy. Using introductory non-computer activities, and problems specifically designed to highlight one of the six cognitive skills, Swan and Black gave their instruction to 133 students in the fourth through the eighth grades. The study used a pretest-posttest single group design, with students pretested for all cognitive skills before the six units and then posttested after the six units. Instruction used class discussion, and individual and paired work on four programming problems for each unit. Investigator constructed outcome measures consisted of word

problems, paper and pencil problem solving activities, the Torrence Test of Creative Thinking, and a constructed test of simple verbal analogies. Swan and Black found significant differences, $p < .001$, between performance on the pretests and posttests for every skill except backward chaining.

The Swan and Black study clearly supports the claims that generalized cognitive skills can be transferred from programming, and that a guided instructional methodology may be necessary to successfully facilitate this transfer. Unfortunately, since Swan and Black did not use a control group in their study, it is difficult to determine how much of the relative pretest-posttest improvement was due to the transfer pedagogy they employed. The programming activity alone, or the initial non-computer discussions of the targeted skills alone, may have been powerful enough to improve the posttest scores. The initial skill introduction is specifically of concern since students received all six units of instruction before being posttested on any of the skills, permitting considerable blending of instruction. Replication is needed, using a control group, to help determine the relative importance of the transfer pedagogy.

In other research looking at guiding transfer of cognitive skills from programming, De Corte and Verschaffel (in press) found that when evidence for the transferable effects of programming was missing, one of two situations were present: 1) students were deficient in necessary programming skills, or 2) an explicit and

systematic orientation for transfer was lacking. These researchers went on to conclude that when positive results were found for the Logo environment, such as with the Clements and Gullo studies, explicit instruction for transfer was typically present in the treatment.

In related research discussion, Delclos, Littlefield and Bransford, describe a "mediational" style of programming which emphasizes this apparent need for explicit orientation for transfer in the instruction. When using this mediational style, a teacher makes specific attempts to frame what is learned in the Logo lesson in a broader context and to bridge specific principles learned to other situations in which the same type of strategy would apply (Delclos, Littlefield & Bransford, 1984). This style involves helping students to see themselves as learners and to become actively involved in analyzing their own thinking strategies. Students are encouraged, often by Socratic dialogue between teacher and students, to generate general principles in their programming activities and to relate these principles to activities in different domains.

Suggestions for guiding the transfer of cognitive skills

When investigating how to facilitate the potential transfer of cognitive skills from programming, it is important to consider that the general "transfer" of cognitive skills between domains has been investigated by a variety of researchers in many fields (Gick & Holyoak, 1983; Cormier & Hagman, 1987; Swan & Black, 1987). One particular

theory for looking at the transfer potential of computer programming has been advanced by Salomon and Perkins (1987). This theory suggests two possible cognitive paths to transfer: 1) low road transfer, and 2) high road transfer.

In low road transfer, near, or same domain transfer, can be achieved by extensive practice which attains an automaticity of a skill. For example, a person learns to drive a car so well by extended practice that other relatively similar cars are driven easily. Thus, a student programmer may become better at specific programming skills within Logo merely by being exposed to a large number of similar programming problems. Yet none of these skills will be of much use outside of the programming domain; transfer will be relatively near, with little far transfer into non-programming domains.

In high road transfer however, more distant, different domain transfer may be achieved. To facilitate such transfer, extensive practice is unnecessary, but fairly extensive concentration and mental abstraction is. In this "high road" to transfer, a person "mindfully abstracts" the skill to be learned so as to see it in a wider more general context involving a variety of domains. Salomon and Perkins indicate that for high road transfer, vigorous and direct instruction emphasizing the general nature of the skill is often necessary to provide a relatively far degree of transfer. Thus, if instructed correctly, a student driving a car may also learn how to drive a truck, by participating in instruction that emphasizes the general principles of driving. A student

programmer then, may improve their general problem solving techniques by being involved in programming instruction that encourages the student to mentally "abstract", or generalize the cognitive skills they use while programming.

Salomon and Perkins analyze the previous Logo research by utilizing this model and indicate that some high road transfer technique was generally used when positive effects for transfer between domains were found. Thus in the Clement study, where positive results were achieved, an instructional scaffolding for high road transfer was employed. In the Pea studies, however, where no positive transfer results were reported, this high road instructional technique was absent.

It is interesting to note that the Salomon and Perkin's model suggests that extensive programming is not strictly necessary for effective high road transfer of a cognitive skill into a non-programming domain; since high road transfer necessitates intensive mental abstraction and concentration, but not intensive practice. This aspect of their model provides that students may not need to be involved in extremely lengthy programming instruction before some transfer between domains is expected. However, this would be true only if the programming instruction is specifically geared toward mindful abstraction of the skill targeted for transfer. Thus, students might be able to attain useful levels of certain cognitive skills through programming instruction without being required to become

programming experts in the process. Salomon and Perkins indicate that their model predicts transfer from even introductory programming instruction, but only when the high road is "forced" by instruction that directly and vigorously helps students to think about programming in the abstract, as a general process incorporating and practicing general cognitive strategies.

Pea, in earlier work summarizing the main conditions needed for general transfer from the programming domain, seemed to incorporate many of the later ideas expressed by both Salomon and Perkins (1987), and Swan and Black (1987). He suggested that to facilitate general transfer, programming instruction would need to: 1) train self management skills, 2) explore the significance of thinking skills, 3) teach the intended skills using real world examples 4) use multiple examples and situations, 5) apply a guided discovery approach in teaching the thinking skills, and 6) provide a favorable motivational environment (1985, p. 2-3). Thus, programming instruction seeking the development of a generalized cognitive skill would probably need to incorporate most of these conditions to create the systematic approach necessary for the explicit teaching of that skill. When these conditions are present, and the instruction "mindfully abstracts" the skill as a general process applicable to other domains, some reasonable hope of transfer could be warranted.

The need for research to test current suggestions for transfer

It is important to note however, that the suggestions for a guided instructional pedagogy to achieve generalized transfer from programming, as discussed by researchers such as Swan and Black, and Salomon and Perkins, are still relatively new in the literature. As with research dealing with the general cognitive benefits to programming, empirical studies attempting to test the effectiveness of the suggestions put forth by these researchers on specific cognitive skills are not yet available. Most of the research suggesting the necessity to "guide" transfer in programming and offering suggestions on how to do so (such as Leron, 1985; Mayer, Bayman, Dyck, 1987; Salomon and Perkins, 1987), is discussion papers rather than empirical studies. Such conjecture is very important, but experimental research is needed to test the validity of these suggestions, and to evaluate the relative importance of the programming component.

Analogical Reasoning and Its Instruction.

Analogical reasoning is one particular cognitive skill that seems especially worthy of any efforts to develop and transfer it. It has been recognized as an important human intellectual skill. In 1956, J.R. Oppenheimer wrote "Analogy is inevitable in human thought" (p. 129). More recently (1982), in his book The Universe Within, Morton Hunt echoed this respect for analogical reasoning and suggested that it was

the most important of all our reasoning processes, encompassing the chief way in which we interpret and deal with the world around us. Both researchers were commenting on the centrality of analogical thinking in daily life; as people continually transfer knowledge from known situations to novel ones. Similar statements, emphasizing the fundamental importance of analogical reasoning, are often found in research regarding this skill (i.e., Holyoak, 1984; Sternberg, 1977b; Moore & Newell, 1973; Newell & Simon, 1972).

What is analogical reasoning?

Analogical reasoning can be defined as the ability to utilize a well understood problem to provide insight and structure for the development of a solution for a less understood problem (Gentner, 1982). Thus, it is this reasoning process which gives us the basic ability to solve a current problem by referencing similarities to previous problems we have encountered. As suggested by Gick and Holyoak, the "essence of analogical thinking is transfer of knowledge from one situation to another by a process of mapping; finding a set of one-one correspondences between aspects of one body of information and aspects of another" (1983, p. 2). Analogical reasoning would seem to be one of the most important skills, if not the most important skill, in general problem solving. George Polya, who has written extensively on problem solving in the mathematics domain, writes: "Inference by

analogy appears to be the most important kind of conclusion, and it is possibly the most important" (1954, p. 43).

Analogical reasoning has also been linked to schema theory. A "schema" can be thought of as an organized body of knowledge conceived theoretically as a set of interconnected propositions centering around a general concept, and linked peripherally with other concepts (Gagne', 1986). This theory, becoming a formally accepted theory of cognitive psychology (Wicks, 1986), indicates that schemas provide internal structure for the assimilation of information in the human mind, and are used as cognitive frameworks for information processing. Analogical reasoning has been said to occupy a generative role in this theory, as a process which creates new schemas (Gentner, 1982; Gentner & Stevens, 1983). New cognitive schemas are theoretically created by the analogical process of deleting differences and identifying similarities between cognitive structures (Gick & Holyoak, 1983).

Component processes in analogical reasoning

As analogical reasoning operates in an individual to transfer and build knowledge, it appears to use specific and identifiable component processes. In extensive research on analogical reasoning, Sternberg has outlined four required component processes that make up the skill (Sternberg, 1977a; Sternberg & Rifkin, 1979). These processes are: 1) encoding, where attributes of concepts are identified,

2) inference, where rules between concepts are discovered, 3) mapping, where a higher order rule relates rules to each other, and 4) application, where a rule is generated from an old concept to a new concept by use of an analogy. Two optional later components have also been included in his theory to encompass the typical structure of multiple choice analogy tests: 5) justification, used to determine the best option when various options are generated in the application component, and 6) response, where a choice selection is actually made to complete the analogy. The component processes are assumed to operate serially, immediately following each other, with movement to the next component facilitated by problem limits or selective attention imposed by the reasoner. Sternberg believes that these components are generalizable to a wide variety of inductive reasoning tasks, especially tasks where the solutions are uncertain.

In a model similar to the Sternberg model of component processes in analogical reasoning, Mulholland, Pellegrino and Glaser (1980), incorporated some of Sternberg's ideas (1977a), with artificial intelligence research by Evans (1968), and with their own modifications to develop a more generalized model of this reasoning. Their model used three general classes of internal processes:

- 1) attribute discovery and encoding processes, where important attributes of each part of the analogy are represented in memory,
- 2) attribute comparison processes, where relationships between analogy parts are inferred, and
- 3) evaluation processes, where the

appropriateness of possible completion items is determined. Specifically, the Mulholland, Pellegrino and Glaser model expanded Sternberg's encoding component and addressed this process in more detail. These researchers felt this modification was necessary due to the relative importance of this component to the rest of the components following it. Validation of the model was based in part on significant relationships found in a 1980 study, using 28 undergraduates exposed to 460 different analogy test items (Mulholland, Pellegrino, & Glaser, 1980).

Standardized tests of analogical reasoning

Component models of analogical reasoning, such as Sternberg's (1977a), and Mulholland, Pellegrino and Glaser's (1980), have extensively used tests with items in an analogy type format such as hammer : nail :: bat : ?. These items often use either linguistic or geometric relationships, and have been included on many standardized psychometric tests. The reliability of such tests is typically high, with intercorrelations between different tests reported as quite high (Guilford, 1967, Ekstrom, French, & Harmon, 1976; Thorndike & Hagen, 1971, 1987). The A:B::C:? item format seems to closely represent the theoretical component processes of analogical reasoning, and it has been suggested by various researchers that these tests are relatively accurate indicators of general analogical reasoning skill (Mulholland, Pellegrino & Glaser, 1980; Greeno, 1978; Sternberg,

1977b). In one particular investigation, incorporating analogies in this format and involving students at Stanford University, Sternberg (1977a), extensively tested and validated various mathematical models for his component processes. Significant relationships were reported supporting the generality of the analogical reasoning process. In a later discussion of this work, Sternberg explicitly commented on the construct validity of tests using the typical analogy format and declared: "The construct validity of performance on tests of analogical reasoning is unimpeachable" (1982, p. 237).

Analogical reasoning in problem solving

Holyoak, building on earlier work by Sternberg (1977a), Hesse (1966), and his own work (Gick & Holyoak, 1980, 1983), has offered a general framework for analogical problem solving using schema theory (1984). He suggested that when a solution for a novel problem (the target), is drawn from a previously solved problem (the base), possibly existing in different domains, four general steps are used. These steps, which are not necessarily implemented in the given order, are:

- 1) mental representations of the base and target problems are constructed,
- 2) relevance of the target problem to the base problem is noticed,
- 3) an initial partial mapping, or set of correspondences, is found between the elements of the two situations, and
- 4) the mapping is extended by retrieving or constructing new knowledge about the problem.

As well as being founded on the work of previous researchers, Holyoak based these steps partially on empirical research from his own studies with Gick (1980, 1983). Four major empirical studies supported his analogical problem solving framework. The first study was used to establish the fact that people can use analogies to generate problem solutions. In this study, 69 students in a high school class were given problems in story format to solve with either first being given an analogous story and solution, or a non-analogous story and solution. Study results found that analogous stories significantly facilitated the students ability to solve the target problem, $p < .001$.

The second study by Gick and Holyoak (1983), attempted to experimentally separate the processes of noticing and applying analogies. This study was similar to the first except for the addition, for some subjects, of a verbal statement included with the initial analogous story that made the underlying principle of the story's solution much more apparent. Such a principle should theoretically be used by the subject in developing a general schema for the particular type of story and solution. Performance between the groups who received analogous stories either with the verbal principle or without the verbal principle was virtually the same. Gick and Holyoak believed that this showed the relative ineffectiveness of incidental verbal statements for general schema refinement in analogical problem solving.

The third study by Gick and Holyoak (1983) investigated factors involved in the degree of transfer from a single base problem to a target problem. In this study, the second study was modified to include incidental diagrams rather than verbal statements. It was thought that incidental pictorial diagrams might be more facilitative to general schema development. However, similar results to the second study were found with no substantial differences between analogous stories and analogous stories with diagrams.

The fourth and final major Gick and Holyoak study (1983), dealt with transfer from multiple analogous stories, emphasizing the possible necessity of providing multiple analogs for generalized schema solution. In this study, 98 subjects were either initially given: 1) two analogous stories, or 2) two non-analogous stories, or 3) one analogous story and one non-analogous story, before being asked to find the solution of a target story. The group given the two analogous stories was most successful in finding the solution to the target problem, $p < .003$, indicating that multiple analogs were the most helpful in the construction a general schema for use in further solutions.

Gick and Holyoak indicated that these four 1983 studies established general problem solving by analogy could take place, and that multiple analogous problems greatly facilitated this problem solving process. In analysis of a learner's ability to recognize that an earlier problem could contribute to a current task, it was apparent from these studies that learners were more likely to recognize prior

experiences as relevant to a new problem when they had developed a more abstract and general schema for sets of problem solutions rather than drawing on specific individual experiences.

In a follow-up study to Gick and Holyoak's fourth study (1983), Mathison and Allen (1987), found that exposure to a single analogous story could yield greater success at determining a solution to a target problem than multiple analogous stories, when a pictorial diagram was included and students were directly instructed within the story to use the diagram. In contrast to the Gick and Holyoak results, Mathison and Allen found that diagrams greatly facilitated analogous solutions when students were directly prompted to use them. Their results suggested that "although multiple similar problem solving experiences may help learners solve new problems analogically, the key variable is not the number of experiences, but the manner in which they are presented and processed" (p. 5).

It would seem apparent from results offered by Gick and Holyoak (1983), and from the follow-up research by Mathison and Allen (1987), that if learners are to be assisted in analogical problem solving, the presentation method will be critical. Learners may need to be initially prompted to notice and use a similar problem, or to notice and use a general principle involving multiple problems.

Rumelhart and Norman (1981) have expressed the importance of such analogical processes in the general teaching for all disciplines. According to these researchers, teaching problem solving should

involve presenting information in a domain where the student is already familiar, and then presenting information in a target domain with a problem that varies only slightly in the number of dimensions, characteristics, or operations. For instance, a child would be taught how to do a fraction problem, by first being presented with diagrams of fractional pies. Such problem solving by referencing better understood problems or problem domains, is very typical of natural real life problem solving (Gentner, 1982; Gentner & Stevens, 1983; Gick & Holyoak, 1983, Rumelhart & Norman, 1983).

The training of analogical reasoning

Although research in the theoretical aspects of analogical reasoning seems fairly well developed, research involving the application of these theories to educational techniques seems much less so. With the exception of studies by White and Alexander (1984), and Alexander et al. (1987), few studies have attempted to actually train analogical reasoning within the dynamic environment of the typical classroom. Most of what has been done, such as work by Sternberg, Ketron, and Powell (1982), has been accomplished in the relatively stable environment of the laboratory. Other studies, like Gick and Holyoak (1983), have investigated general processes underlying analogical reasoning, but have offered only peripheral suggestions for classroom application.

Many college critical and creative thinking courses, however, still incorporate some instruction in analogies (Halpern, 1987). This may be due in part to the poor performance of even college aged adults on general reasoning tasks where analogical type reasoning is incorporated. Research by McKinnon and Renner (1971) found that only 25% of first-year college students scored at the Formal Level of thought on Piagetian tests (1971). Other researchers, such as Nummedal, have stated concern that most adults are functionally unable to use formal reasoning processes by indicating that studies in the area indicate that "Less than 50% of the students in our universities are able to use formal reasoning processes confidently and reliably" (1987, p. 87). In light of such research, there would seem to be an apparent need to provide some general analogical reasoning training within our educational system.

Unfortunately, due to the lack of classroom research, it is unclear whether laboratory implications of analogical reasoning can be applied successfully to the typical classroom environment. It would appear that to successfully apply laboratory established theories to classroom applications, some systematic method to do so may be necessary. Training in analogical reasoning, like other cognitive skills, should probably be systematic and deliberate if it is expected to transfer across domains for general use.

In research attempting to train analogical reasoning in the classroom, and to apply laboratory established principles to the

classroom environment, Alexander, White, Haensly, and Crimmins-Jeanes (1987) tried the direct instruction of analogies with fourth graders. They had piloted the study a few years earlier (White & Alexander, 1984). Within their recent study, these researchers used Sternberg's four mandatory components of analogical reasoning as an instructional framework for direct lessons in analogies, and then extended the process to reading comprehension tasks. Their training involved two phases. Phase one was an initial training phase where students received intensive instruction in the component processes of analogical reasoning. This phase used an instructional approach which permitted varied practice and classroom discussion, incorporating student verbalizations of each component as they were used on problems. The treatment in this phase consisted of one 50 minute session for each of the four components. The second phase of instruction consisted of six weeks of intermittent training involving general application of the four components. Training during this phase also sought to incorporate stories as well as standard word analogies. Significant effects of training were found, with implications of near transfer given by significant results on verbal analogies of the Woodcock Reading Mastery Test, $p < .002$. However, significant effects were not found for reading tasks of the Comprehension Inventory, $p > .05$, implicating that no far transfer had occurred.

In a second follow-up study to look at the influence of age and ability on their results (1987), Alexander, White, Haensly, and

Crimmins-Jeanes repeated their first study with 34 eighth graders enrolled in an honors language arts class, and 96 tenth graders enrolled in a non-honors English class. Positive effects for group, $p < .001$, and for grade, $p < .002$, were found within the analysis.

Alexander, White, Haensly, and Crimmins-Jeanes, concluded that these two studies demonstrated that "componential analogical reasoning training could be provided in the uncertain and dynamic context of the classroom" (1987, p. 401). Also, they believed their work demonstrated that analogical reasoning training could be successfully provided to older students, and to students of gifted and average abilities. These researchers were careful to note however, that analogical reasoning is a rather complex and global skill, and that speculations regarding the impact of their training on overall analogical reasoning ability should remain cautious.

The need to consider possible interactions

Interpretation of the effects of a particular treatment, such as analogical reasoning training, operating with a particular set of students often warrants caution. In educational research it is important to consider that the typical classroom usually consists of a heterogeneous group of students displaying a variety of characteristics. Each student may bring to the learning environment a unique set of attributes, learning styles, relative abilities, and past achievements.

The potential for interactive effects of student characteristics with instructional treatment is well established in educational research (see Cronbach & Snow, 1977; Snow 1977; and Holtan, 1982). Often, student characteristics with a potential for interaction can be considered as one of three general types: 1) student ability, 2) student aptitude, or 3) general student attributes. However, some overlap of this terminology is often present in the literature (Federico, 1978).

Student ability, as used in treatment interaction research, is often associated with a relative level of a particular cognitive skill, such as spatial visualization, verbal fluency, etc. However, many researchers have settled on the construct of "general ability" in looking at differential effects of an instructional treatment across student subgroups (Cronbach & Snow, 1977). More recently, "general ability" has been subdivided into "crystalized ability", representing cognitive skills applied to a more familiar situation, and "fluid ability", representing cognitive skills applied to a newer less known situation (see Snow, 1980; Hart, 1986). High crystalized ability would indicate efficiency in the routine application of stored knowledge, where new tasks are quite similar to previous ones. High fluid ability, however, would indicate efficiency in adapting this stored knowledge, often spontaneously, to tasks substantially different than ones encountered in the past. The possible interactive effect of such relative student ability, with a specific instructional treatment, is one of the most consistent

interactions in educational research (Wittrock, 1974; Anderson, 1982; Snow and Lohman, 1984).

In contrast to student ability, which focuses on a relative level of cognitive skill, student aptitudes give more emphasis to the actual form and structure of cognition. Student aptitudes usually consist of differing cognitive styles or learning preferences (Federico, 1978). Student aptitudes also have been shown to have a potential interactive effect with instructional treatment. For example, student "field independence" or "field dependence" has been shown to have general interactive effects in the learning of mathematics (McLeod and Adams, 1980; McLeod and Briggs, 1980). Other examples of student aptitudes thought to have possible interactive effects are locus of control, brain hemisphericity, and cognitive complexity (McLeod, 1979; De Leeuw, 1983).

Student attributes, rather than being associated with level or structure of cognitive processing, typically refer to the more fundamental characteristics of a student, such as gender and age. The potential for age related differences in cognitive processing has been well established (Goldman, Pellegrino, Parseghian, & Sallis, 1982; Goldman & Bisanz 1980; Lawson, 1982, Sternberg & Downing, 1982), and research considering gender differences in the learning of mathematics has been evolving for some time (Schildkamp-Kundiger, 1982). In addition, general past achievement has been shown to have consistent interactive effects with cognitive instruction (Tobias, 1976).

It seems apparent that research investigating the classroom training of analogical reasoning must also carefully consider the possible interactive effects of individual student characteristics. When being instructed in such a global and complex problem solving skill as analogical reasoning, students with varying characteristics may react quite differently to an instructional treatment.

The need for additional research in classroom instruction

Although research involving the processes of analogical reasoning is fairly well developed, research involving the application of these processes to instruction and training in the classroom is in a relative infancy. There is substantial work to be done before analogical reasoning instruction can be incorporated systematically and effectively into the classroom. As suggested by Alexander, White, Haensly, and Crimmins-Jeanes (1987), regarding further research needed in their analogical reasoning based instructional work:

"It is for future research efforts to support the conclusions of this study by demonstrating further the facilitative effects of classroom-based, componential training in analogical reasoning. It may prove useful for future studies to examine alternative near and far transfer tasks, to test the effectiveness of analogy training embedded in different content areas such as

mathematics or science, or to provide such training to students younger or older than those in the present study" (p. 402).

When considering the research reviewed regarding analogical reasoning, strong implications for specific additional studies become apparent. Research in this area is at a point where investigation concerned with the actual classroom implementations of more fundamental research is imperative. Research is needed that tests applications of componential analogical reasoning training in the classroom environments of various disciplines, and with a diversity of students. A study incorporating analogical reasoning training within the domain of computer programming, examining near transfer within the programming domain, and far transfer beyond the programming domain, may be just such a needed study. However, it is first necessary to consider how analogical reasoning typically operates in the computer programming domain.

The Relationship of Analogical Reasoning to Programming

As discussed by Pea and Kurland (1984a), computer programming is indeed a complex task involving many essential skills. Analogical reasoning has been suggested by researchers as one of the most important of these skills; fundamental in the programming activity

whenever a programmer seeks to recognize and exploit the similarity of different programming tasks (i.e., Kurland, Clement, Mawby, & Pea, 1987; Mann, 1986; Pennington, 1982, Sneiderman 1976). Many of these researchers focus on the contrast between expert and novice programmers when discussing how analogical reasoning typically operates within the programming domain.

Analogical reasoning and expert and novice programmers

Analogical reasoning has been linked to programming by investigations of how expert programmers generally approach programming problems. When confronted with a new programming problem, an "expert" programmer will often tend to perform an internal search for programming problems of a similar nature that they have encountered in the past (Brooks, 1977). These earlier problems, and their respective solutions, provide the expert with an internal "storehouse" of general sub-routines and structures potentially useful in developing or determining a solution to the new problem. The expert looks at similarities between the problems, and eventually attempts to transfer a solution framework from one problem to the other. This analysis of something new in their programming experience, by utilization of something old in their programming experience, is essentially the process of analogical reasoning as defined by Gentner (1982).

In contrast, novice programmers tend to focus on the lower level aspects of a programming activity, such as the syntactical structure of the program code (Sneiderman, 1976; Sneiderman & Mayer, 1979; Onorato & Schvaneveldt, 1986). In a study by Adelson (1981) five novice programmers and five expert programmers were given 16 randomly shuffled lines of programming code and asked to memorize as many lines as possible after being shown each line for a 20 second interval. In comparing the recalled lines of the novices and experts, it was found that the experts not only recalled significantly more lines than novices, but also grouped these lines into meaningful "chunks" which related to procedural or functional aspects of the program. It was believed that experts were using internal representations of previously encoded programs to meaningfully group the newly memorized lines of code. The novices, however, used smaller less meaningful chunks, and organized these around more superficial syntactic aspects of the code. Novices, unlike the more successful experts, seemed unable to draw analogies to the more functional and structural aspects of programs, to assist in their mental grouping of the lines.

Such differences in cognitive behavior have also been observed in program debugging attempts by novices and experts. In two studies by Gugerty and Olson (1986), novices and experts were asked to debug three Logo programs and three Pascal programs having a single bug in each. Using transcripts of the online debugging experience, experts

were found to correct the bugs much more successfully than the novices by use of carefully formulated hypotheses about where the bug might be, as determined from symptoms of the output. Novices, despite spending as much time as experts in thoughtful analysis of the program, were unable to create quality hypotheses regarding the possible location of the bug. In addition, they often added new bugs to the program when attempting to correct the original bug. Since experts and novices distributed their debugging activities in about the same overall way during the activity, it was believed by these researchers that a qualitative cognitive difference between the experts and the novices was being observed. They speculated that the experts were more efficient at the inherent encoding process used in forming successful hypotheses of bug location, and could draw on a large library of symptom-bug associations in forming specific hypotheses. Such a systematic mapping process, as exhibited by the experts, would fit the analogical reasoning definition of Gentner (1982).

Analogical reasoning through programming

Based partially on the analogical reasoning behavior of programming experts, researchers are beginning to target analogical reasoning as a specific cognitive skill with the potential for direct practice and development in computer programming. In a correlational study, Clement, Kurland, Mawby, and Pea (1986) looked specifically at analogical reasoning and the reuse of subprocedures in

Logo programs of seventeen ninth, tenth, and eleventh grade females. Using a programming task developed in an earlier study (Mawby, 1984), these researchers attempted to correlate various aspects of the student programs with success on a Gick and Holyoak analogical reasoning task. Outcome variables on the programming task involved scores for programs correct, number of commands used, correct use of subprocedures, correct use of repeat statements, correct use of variables, and reuse of subprocedures across programs. The only significant correlation to success on the analogical reasoning task was the reuse of subprocedures across programming problems, $p < .01$.

Clement, Kurland, Mawby, and Pea felt that since reused subprocedures implied a structure mapping between programming problems, this variable was probably the most indicative outcome variable of student analogical thinking within the programming activity. Encouraged by results isolating the reuse of subprocedures variable as the only significant correlation to the analogical reasoning task, and the inherent nature of analogical reasoning within the programming process, these researchers felt that "analogical reasoning could develop through programming because it underlies certain programming practices and because programming involves a focus on abstract structural relations" (p. 484). They were careful to emphasize the correlational nature of their study however, and that the complexity of these relationships implied that the transfer of analogical reasoning

skill, as well as other cognitive skills from programming, may need to be directly supported, rather than expected to occur spontaneously.

Swan and Black, in their investigation of the cross-contextual transfer of problem solving skills (1987), included the investigation of analogs as one of the six problem solving skills that they targeted for improvement by guided Logo programming instruction. These researchers believe that the analogical reasoning process is inherent in a programmer's analysis and refinement of computer code, as the code is systematically modified by the programmer based on the output it produces when implemented by the computer. As suggested by Swan and Black: "we believe programming environments inherently support the development of analogy, in that one is always mapping between computer code and program output" (1987, p. 8). They found a significant difference for improvement on an investigator constructed analogies test, of the form Gun:Bullet::Bat:?, $p < .01$. This was true whether program output was in the form of words and lists, turtle graphics, or a combination of the two types of output. However, as discussed earlier in this chapter, it is difficult to say how significant a role the activity of programming played in the actual development of the demonstrated analogy skill; as the guided instruction methodology employed substantial off computer tasks, focused on solving analogies, before the programming activity began. Swan and Black did not attempt to investigate this contingency, which would have necessitated the use of a control group.

While Swan and Black looked at analogical reasoning as one problem solving skill of six targeted, and focused on the design of the guided instruction, a study by Mann investigated analogical reasoning skill more directly as a potential benefit to programming (1986). Matching by sex, and scores on a pretest of the outcome measure, Mann used two groups of eighth grade students to participate in the ten week study. The experimental group received instruction in Logo focused on eight modules ranging from the use of primitives to recursion. The control group did not use Logo at all, but was involved in word processing and computer assisted instruction in the content areas of reading, math, english, and science. Analogical reasoning skill was evaluated by use of the Cognitive Abilities Test-Nonverbal Battery. Mann found a significant positive effect for the Logo treatment, $p < .05$. No differential gains between males and females were observed.

Mann's results encouraged him to suggest that "Logo may be influential in facilitating the development of analogical problem solving strategies" (p. 76). Mann investigated programming and analogical reasoning using Logo in a fairly traditional Logo programming environment with extensive experimentation combined with teacher and student generated goals. No specific instructional guidance was incorporated to target the skill of analogical reasoning; it was merely investigated as a potential outcome to the general programming instruction. For this reason, Mann suggested that further investigations were needed, perhaps with different age groups and

ability levels, but especially research using more "guided" programming instruction, which explicitly incorporated and targeted the skill of analogical reasoning in the programming environment.

Analogical reasoning components through programming

As previously discussed, analogical reasoning has been investigated as a skill which can be broken into specific component processes (Sternberg, 1977a,b; Mulholland, Pellegrino & Glaser, 1980). Sternberg proposed the processes of encoding, inferring, mapping, and applying as mandatory components in the general skill of analogical reasoning. The Logo programming language uses a structure which can potentially support the development of component skills, such as the analogical reasoning components described by Sternberg. The modular nature of the language makes it especially conducive to the explicit practice of the component processes involved in many cognitive skills. Clements and Merriman (1987), in their recent discussion of componential developments in Logo programming environments, describe how children programming in Logo might employ component processes:

"Children who begin a Logo project may start by making a drawing, (their problem goal). They might selectively encode parts of that drawing to determine basic shapes that can be disembedded and constructed as procedures. In

addition, they might encode the salient properties of shapes and relationships among shapes. They then might selectively compare their present problems with past procedures to determine if these old procedures and the methods used to construct them might assist in solving problems at hand. Children also might selectively combine procedures to create numerous figures from a limited number of components and, more importantly, combining parts of a problem solution into a unified whole" (p. 4).

Analogical reasoning and the programming environment indeed seem well intertwined. As old problems are searched to provide insight into the solution to new problems, the various component processes of the skill (such as Sternberg's encoding, inferring, mapping and applying), appear to be in genuine operation. As stated by Clements and Merriman: "It becomes apparent that the Logo environment could serve as a vehicle for the development of componential skills" (1987, p. 8). There would seem to be implication that analogical reasoning could be fostered by Logo programming carefully targeted at the component processes of the skill.

Summary

The purpose of this chapter was to review previous research related to Logo programming instruction and the potential development and transfer of analogical reasoning skill. Two main bodies of research were tapped in this literature review: 1) research involving the development of cognitive skills from computer programming, and 2) research involving the training of analogical reasoning. Research in each of these two areas has developed relatively separately.

Research involving the development of general cognitive skills from computer programming has been an area where anecdotal observations and contemplative discourse has dominated. These observations have tended to suggest that computer programming offers a potential for the development of general cognitive skills, but that such development may rest substantially on the instructional environment included in the programming activities. The necessity of further research, preferably of an empirical and focused nature, has been declared by many researchers (i.e., Pea, 1983; Khayrallah & Maud Van Den Meiraker; 1987; Walker, 1987).

Research involving the classroom training of analogical reasoning is in early stages even though, somewhat ironically, theoretical research on the skill itself has existed for some time. Researchers, based on laboratory investigation, have broken the skill into various

component processes which may offer a useful and practical framework for classroom training. The need for further research, actually attempting to train analogical reasoning in the classroom, and incorporating it into a variety of content areas has been suggested (Newby & Stepich, 1987; Alexander, White, Haensly, & Crimmins-Jeanes, 1987; Mathison & Allen, 1987).

It seems from a review of the research investigating the development of cognitive skills from programming, and the research investigating the classroom training of analogical reasoning, that these two research areas can act as a catalyst to each other. Research investigating the development of cognitive skills from programming has been weakened by a lack of empirical studies focusing on specific cognitive skills. Analogical reasoning, with its extensive theoretical foundation and fundamental link to the programming process, is one cognitive skill worthy of specific targeting in programming studies. Similarly, research investigating the training of analogical reasoning is in its infancy, requiring further research in a variety of classroom settings and content areas. Computer programming, with its explicit nature and inherent use of analogical problem solving, is one content area that seems unusually appropriate for the investigation of such training.

It would seem that a necessary research step toward the understanding of the potential cognitive benefits of computer programming, and toward the identification of potential instructional

methods for the classroom training of analogical reasoning, would be a study attempting to instruct analogical reasoning by use of classroom programming activities. Studies of this type would help overcome limitations to research in both areas, and help narrow the gap that often exists between educational research and classroom practice. Such a study is described in Chapter Three.

CHAPTER III: METHODS AND PROCEDURES

This study investigated the potential of guided Logo programming instruction for use in the development and transfer of general analogical reasoning skill. This investigation, focusing on analogical reasoning as one specific skill, and guided Logo programming as one particular method, should contribute to the continual search for possible methods to instruct general cognitive skills.

This study incorporated research from two distinct areas:

- 1) research on the cognitive benefits to computer programming, and
- 2) research on analogical reasoning. It was determined that a specific study attempting to develop analogical reasoning through guided computer programming would help meet research needs in both of these areas.

In investigating the potential of guided Logo programming instruction for use in the development and transfer of analogical reasoning, two particular effects of the instruction were considered worthy of analysis. These effects were: 1) the far transfer effects of the instruction, as measured by a standardized test associated with general analogical reasoning, and 2) the near transfer effects of the instruction, as measured by the reuse of subprocedures on a constructed test of programming problems.

This chapter will discuss the methods and procedures used to investigate these effects. The chapter was divided into five main sections: 1) the sample of subjects used in the study, 2) the

instructional treatments developed for the study, 3) the research instruments used to empirically investigate the effectiveness of these treatments, 4) the research design and procedures used in conducting the study, and 5) the directional hypotheses and analysis of data procedures. These sections summarize the methodology that was incorporated to investigate the potential of guided Logo programming for use in the development and transfer of analogical reasoning.

Subjects

The subjects used in this study were students enrolled in the Fall 1988 class of Secondary Education 101, at Iowa State University. This class was titled "Educational Applications of Computers", and was an educational computing class designed to offer students in the teacher education program an opportunity to experience and to gain confidence in the use of computer technology as it can be applied to education. The course outline included six major topics: wordprocessing, Logo programming, databases, spreadsheets, desktop publishing, and mainframe computer use. The Logo programming unit, of three week duration, was used for the study.

The class had 171 students initially enrolled, with 144 of those students included in the study. A total of 27 students were removed from the sample for either failing to attend the class, or for failing to consistently attend the portion of the class used for the treatment. As

a criteria for removal, students were allowed to be absent from only one instructional period within the three week Logo unit. The 27 subjects removed from the study missed two or more periods within that unit.

Students in the sample were given a brief questionnaire to investigate the homogeneity of their backgrounds. The questionnaire has been given at the beginning of the course to help prepare yearly revisions to course content. It included items asking for college major, computer programming experience, computer access, mathematics course experience, computer anxiety, current grade point average, age, year in school, and gender (see Appendix A).

College major of subjects

The majority of the students enrolled in the course are usually elementary education majors. The entry questionnaire confirmed this, as 66% of the students listed elementary education as their major. The remainder of the sample included 6% of the students who listed secondary education, 9% who listed physical education, 8% who listed agricultural studies, and 11% who listed non-teacher education or agricultural majors such as business. The substantial number of non-teacher education majors was common for the course, as it drew a significant number of students from other colleges who were interested in taking the course for the experience it provided in

microcomputer based wordprocessing, database, and spreadsheet applications.

Age, year, and gender of subjects

The mean age of the students was 21.3 years, with a range of ages from 17 to 42. The majority of the students were of typical college age, however, with 74% of the students between the ages of 18 and 21, inclusively. A relatively equal split between each college class was represented in the sample. The questionnaire data indicated that 31% of the students listed themselves as freshman, 22% listed themselves as sophomores, 26% listed themselves as juniors, and 22% listed themselves as seniors. The gender breakdown of the sample was 25% male and 75% female. The elementary education majors had the lowest male to female ratio, with only 12 males to 84 females.

Programming experience of subjects

The programming experience of the sample was quite limited. The initial survey indicated that 78% of the students had not completed a single programming class in college, and 18% had completed only one class. The students who listed one class, as well as the 4% of the sample who listed more than one class, commonly listed classes which were probably not strictly programming classes, such as a computers in business course. High school programming experience was also fairly limited. Questionnaire data indicated that 44% of the

students listed no high school programming course, and 47% listed only one course. Generally, students reported a single course in BASIC programming as their only high school programming experience, if they had a programming course in high school. Students who listed more than one course, accounting for 9% of the sample, also commonly listed high school courses which were probably not strictly programming courses, such as a business machines course.

Math background of subjects

The math background of the students in the sample was very limited. Many of the students, 41%, had not yet had a college math course. For the 31% of the sample who listed a single math course, the course listed was usually the low level math course required for elementary education majors. The students who listed two math courses, accounting for 22% of the sample, often listed college algebra along with this elementary education math course. Only 6% of the sample reported having taken three or more college math courses, with only four students listing any math course above college algebra.

Computer access and nervousness of subjects

From the survey it was apparent that for many of the students, using a computer was still a new experience for them. A majority of the students in the sample, 69%, indicated that they did not have access to a computer outside of the university facilities, and only 11%

indicated that they had worked with a computer at all outside of their high-school and college course work. In response to the question about how nervous they felt about their upcoming computer experiences in the course, 13% of the students responded they were very nervous, 26% responded that they were somewhat nervous, 41% felt neither nervous or confident, 16% felt somewhat confident, and 4% felt very confident.

Summary of sample characteristics

In summary, the majority of the students in the sample were female elementary education majors. However, a small but significant number of students were male agricultural education majors, with a variety of other majors represented in the sample. Most of the students in the sample had relatively little experience in computer programming, and had taken very little mathematics at the college level. Students in the sample also reported very little experience with computers outside of their academic work, and a little less than half of the students in the sample expressed some degree of nervousness about their upcoming computer experiences.

Treatment Groups

In order to investigate the potential of guided Logo programming instruction for use in the development and transfer of analogical

reasoning, the sample was broken into two distinct treatment groups. The experimental group was involved in programming instruction that systematically guided students through a structured process of analogical reasoning, and encouraged students to use this process in the planning of their programs. The control group used a more traditional approach to Logo programming instruction, and gave students a large degree of freedom in the planning of their programs.

To help ensure that both treatment groups received the same instructional content, and that only the instructional methodology itself was varied, both groups worked through the same set of in-class problems and examples. Programming problems were presented to students in both groups on prepared notesheets roughly structured along Polya's four steps to problem solving. This approach was used to remind students and teachers of the sequence of instruction for their respective groups. Such worksheets were also thought to be helpful in preventing "overloading" the students with the combined cognitive demands of both the instructional content and instructional strategy, especially in the experimental group. The use of such structured notesheets is suggested by Perkins, Simmons, and Tishman, as a possible way to help students manage the additional cognitive load of instructional strategies (1989, pp. 13-16).

Content ranged successively from simple procedures and turtle graphics, to more difficult problems utilizing multiple variables and recursion. As a resource for course content, the textbook, LogoWorks:

Lessons in Logo, by the Terrapin Logo Company was used to provide examples and problems. Content sheets for the experimental and control groups are given in Appendices B, C, D, and E.

The experimental group: guided Logo programming instruction

The guided Logo programming group acted as the experimental group for the study. For this group, initial program planning was structured to explicitly emphasize the skill of analogical reasoning. This involved stepping students through a careful problem solving strategy that analyzed a previously completed programming problem to help find insight into the solution of a new programming problem. The experimental group was required to use this particular strategy whenever they planned a program to accomplish some output.

This instructional treatment, which involved programming activity focused on the general development and transfer of analogical reasoning, sought to incorporate the three pedagogical elements suggested by Swan and Black as common to studies reporting positive results of cognitive skill transfer from Logo programming (1987, pp. 6-7). The three transfer elements suggested by Swan and Black were that the instruction: 1) focus on specific problem solving aspects or skills, 2) directly instruct the targeted skills, and 3) utilize a mediated learning approach to student and teacher interaction. Each of these components was explicitly addressed in the instruction of the experimental treatment group. This approach helped ensure that the

instruction targeted the development of a generalized analogical reasoning skill, rather than a skill specific only to the computer programming domain.

First transfer element In meeting Swan and Black's first pedagogical transfer element, a focus on specific aspects of the problem solving process, the experimental group emphasized the analogical reasoning process inherent in programming, rather than the programming activity itself. To help provide this reasoning emphasis, students initially spent about 15 to 20 minutes working through the analogical reasoning planning activity before ever turning on the computer. The instructor would lead students through this planning process by active class discussion, or by walking around and ensuring that everyone was actively engaged in the step-by-step process as it was outlined on the instructional worksheets. Students were reminded that they were using the computer programming activities as a way to practice the general analogical reasoning process, with the learning of this general reasoning strategy as an important goal in their programming instruction. Thus, students were practicing and focusing on a specific problem solving process or reasoning skill, but were applying it where it would be currently most useful to them, during the initial planning of an assigned program.

Second transfer element To meet Swan and Black's second pedagogical element and directly instruct the skill of analogical reasoning, Sternberg's four main component processes of analogical

reasoning (1977a) were used as a framework for planning the student programs (see Appendix F). The components of encoding, inferring, mapping, and applying, were directly incorporated into structured the discussions which were used with students to help initially plan their programs (see Appendices B and D). This approach resembled the process used in the work by Alexander, White, Haensly, and Crimmins-Jeanes, with linguistic analogies (1987). In their study, instruction was designed so that students systematically worked through each component as they planned programs to give some desired output. The turtle graphic capability of Logo made it especially conducive to providing the opportunity for well-structured and focused programming examples of these component processes. Using the A:B::C:D analogy format, the programming activities were structured by the pairing of program output with corresponding program text. Students were prompted, by use of the four components, to actively use the graphical output and program text of a previously solved problem to find insight into developing the text of a new programming problem. The use of the Sternberg Component processes carefully structured the cognitive tasks of writing the new program, and met the definition of direct instruction as expressed by Doyle (1983).

Third transfer element To provide a mediational approach to interactions by teacher and students, and to meet Swan and Black's third pedagogical element to encourage transfer, a class discussion style was patterned after the suggestions of Delclos, Littlefield, and

Bransford (1984). In their work, they suggested that in a mediational style of teaching Logo the teacher should make "conscious attempts to frame what is learned in the Logo lesson in a broad context and to bridge specific principles learned to other situations where the same type of strategy would apply" (p. 9). Within the experimental treatment, students were periodically given other examples illustrating the use of the Sternberg analogical reasoning components, and discussed how the reasoning process was similar to what they were using in the programming domain. The teachers continually tried to emphasize to students that they were using the skill of analogical reasoning to help solve problems in computer programming, but that the skill could also help them solve problems in other areas. Additionally, when students asked specific questions about errors in their programs, teachers tried to respond in a Socratic type format, referencing the analogical reasoning process whenever possible.

The control group: traditional Logo programming instruction

A group involved in programming instruction incorporating a more exploratory and traditional type Logo environment was used as the control group in the study. In contrast to the experimental group, the control group instruction did not focus directly on the cognitive skill of analogical reasoning. Instead, the instruction provided as much student freedom as possible in planning of the solution to each assigned programming problem. As with the experimental group,

students in the control group were given programming problems which were specified on notesheets. Polya's four steps to problem solving were used as a structure for the notesheets, to help teachers and students sequence instruction (see Appendices C and E). The control group, however, did not use the Sternberg component processes in the planning stage of a program. At this point in the instruction, control group students were told to write a program to accomplish a specific output, and that previous work might or might not be helpful.

Teachers working with the control group, as with the experimental group, were careful to try to maintain, as much as possible, a traditional Logo programming environment while students worked at the computer. Teachers avoided stepping in to help students unless they specifically asked for assistance, and then tried to assist students only by asking them small Socratic questions rather than by giving them specific answers.

The control group instructional sequence provided that students in this group were on the computer approximately twice as long as those in the experimental group. During the time that students in the experimental group were planning their solutions by use of the analogical reasoning components, students in the control group were actually trying to program and experiment with their solution on the computer. This additional exploration time was thought to be conducive to the more traditional Logo approach.

Summary of study treatments

In order to investigate the potential of guided Logo programming instruction for use in the development and transfer of analogical reasoning, it was necessary to use two distinct treatment groups. The experimental treatment group used a methodology that sought to focus on the skill of analogical reasoning and encourage its general transfer. This instruction relied heavily upon the suggestions of Swan and Black (1987), for the effective transfer of specific cognitive skills. To meet the requirements of these suggestions, the experimental treatment used the Sternberg component processes of encoding, inferring, mapping, and applying, to assist in directly instructing analogical reasoning while students were planning their programs. In contrast, the instruction for the control treatment group did not actively emphasize the skill of analogical reasoning, and sought to allow students as much freedom as possible in planning and testing their own solutions to the programming problems. Finally, the instructional content was carefully controlled for both groups, so that the only difference between the groups was the instructional methodology employed in delivering that content.

Research Instruments

Three instruments were used to provide measures of dependent variables within the study. To evaluate the differences between the

groups in general analogical reasoning, the Cognitive Ability Test - Nonverbal Battery, published by the RiverSide Publishing Company, was used as an operational measure. To evaluate differences between treatment groups in the reuse of program subprocedures, a programming test developed and used by researchers Kurland, Clement, Mawby, and Pea (1987; also Clement, Kurland, Mawby, & Pea, 1986) was slightly modified and used in the study. Additionally, to determine if overall comprehension of the LogoWriter language had been kept similar between treatment groups, and to provide insight into results on the reuse of subprocedures instrument, a LogoWriter basic comprehension test, in a multiple choice format, was also created and locally standardized.

The Cognitive Ability Test

This test was used to provide an operational measure of the general analogical reasoning ability of the subjects. It was developed by Thorndike and Hagen and published by the RiverSide Publishing Company. In this study, the Non-Verbal Battery of Form 4 - Level H, was used to compare treatment groups.

The Nonverbal battery is composed of three separate subtests that involve geometric type problems, presented in three different formats, developed to reflect the problem solving behaviors in general analogical reasoning. The three subtests are Figure Classification, Figure Analogies, and Figure Analysis. The first subtest, the Figure

Classification test, required students to determine what features three geometric figures have in common and to select an additional figure with the same common features from five different response options. The second subtest, the Figure Analogies test, required students to complete a geometric analogy of the $A:B::C:?$ form by determining the relationship between two geometric figures, and then selecting one of five options that duplicated the relationship between a third figure and the selected figure. The third subtest, the Figure Analysis test, required students to mentally fold a piece of paper and then punch it with a given number of holes in a specified location. Students were then required to select the response option that indicated how the paper would look when it was unfolded. The score on each subtest was determined by finding the number of correct responses for each subtest. An overall composite score for the battery was computed by a simple summation of these scores.

The Cognitive Ability Test has been extensively standardized, and has a KR-20 reliability index of .91 for the Nonverbal Battery, as indicated by the Preliminary Examiners' Manual (Thorndike & Hagen, 1987a). It is a commonly used test in research dealing with cognitive development, and has been used effectively in studies investigating analogical reasoning. Specifically, it was incorporated by Mulholland, Pellegrino, and Glaser (1980) in their problem solving research of geometric analogies and used to validate their model of analogical

reasoning. It was also utilized by Mann (1986) as an instrument to examine the problem solving effects of Logo programming.

The Nonverbal Battery was considered appropriate for use in this study due to the geometric nature of student output when using the turtle graphic features of Logo. Since students would be involved in programming problems that incorporated graphical output, an operational measure of analogical reasoning that used graphical rather than linguistic problems seemed the most conducive to observing differences in treatment effects. It was assumed that scores on such a test would give an indication of analogical reasoning skill that had developed through the programming activities, but was not limited to the programming domain.

The reuse of subprocedures programming test

This test was a slightly modified version of a programming proficiency test developed at Bank Street College by Kurland, Clement, Mawby, and Pea (1987). It was originally developed to examine three aspects of programming proficiency: reusability of code, flow of control, and program decomposition. In designing the test, these researchers were less interested in testing the knowledge of individual commands, and more interested with assessing the comprehension of the overall structure of the language and the pragmatics of programming. The test originally consisted of five programming problems, with two distractor problems, and was given as a thirty

minute paper and pencil test where students wrote programs for five of the seven test problems.

In this study, the test was modified slightly, so as to more directly compare student reuse of subprocedures between programming problems. In adjusting the test, the two distractor items were removed and the order of the problems fixed. The two distractor items were removed because these output figures could not be broken down into effective subprocedures. This modification also provided that a more manageable number of problems, five rather than seven, remained in the test (see Appendix K). The order of these remaining problems were fixed by requiring students to attempt the five problems in a specific order. This modification helped provide that students would be compared on essentially the same task, with the same problems in the same order. Additionally, to increase the overall number of possible subprocedures, two of the output figures were shaded, and three of the output figures were slightly enlarged, so that two additional general subprocedures might be written by the students: one that would shade figures, and one that would initially place figures to avoid screen wrap around.

The test was administered in an on-line rather than paper and pencil format to provide a more realistic and dynamic programming environment. This online format also aided in securing data, as programmed solutions could be saved on disk, checked by computer execution, and analyzed from computer printouts; a far easier task than

trying to decipher a large number of student handwritten solutions. The time limit was also extended to 60 minutes to provide students with sufficient time to test and refine their solutions. Thus, except for the extended online format, and the slight modifications necessary to provide a more focused look at the student reuse of subprocedures, the programming test was basically the same as that designed by the Bank Street College researchers and used in their studies (Kurland, Clement, Mawby, & Pea, 1987; Clement, Kurland, Mawby, and Pea, 1986).

To score the reuse of subprocedures on the test, similar scoring procedures were used to those suggested by Kurland, Clement, Mawby, and Pea (1987). First, subprocedures that were used in more than one of the five problems were identified and listed. Then for each of these reused subprocedures, a count was made of the number of problems in which it was used. This process gave a number from two to five for each of the listed subprocedures. The problem counts for each reused subprocedure were then added up to create a total score. This total score was assumed to represent the reuse of subprocedures for a particular student.

The test targeted five types of potential subprocedures that might be used by the students. They were: 1) a general rectangle procedure that could be used in problems A, D, and E, 2) a general square procedure that could be used in problems B, and D, 3) a general shading procedure that could be used in problems C and E, 4) a

general horizontal and vertical move procedure, useable in problems A, B, C, and E, and 5) a general initial placement procedure that would need to be used in problems B, C, and E, to prevent screen wrap-around. This plan provided that a student's reuse of subprocedure score might range from zero to fourteen. Additionally, it was foreseen that a student might write a general subprocedure for a rectangle and also use this procedure to produce a square. In that situation, the rectangle procedure would be tallied as being used in the observed two to five problems, and the square procedure in none.

LogoWriter basic comprehension test

To determine if general comprehension of the LogoWriter language had been consistent between instructional treatments, a LogoWriter basic comprehension test was developed for the study. This instrument tested students on the fundamental commands and concepts in the LogoWriter language that acted as the basic programming content for the instructional treatments. It was decided that such an instrument was especially needed to accurately interpret eventual results on the reuse of subprocedures programming instrument; results that might be heavily influenced by lower level differences in basic understanding of the LogoWriter language. Since both instructional treatments were to vary only in the delivery of the content, and not in the content itself, this instrument also helped to verify that both groups learned at least a minimum level of LogoWriter

commands and concepts. Thus, to help determine that any observed reuse of subprocedure differences were indeed due to treatment effects, and not to lower level comprehension differences, a 30 question multiple choice basic comprehension test was developed and locally standardized (see Appendix L). It was determined from an extensive literature search, and by correspondence with numerous researchers, that no other suitable, previously developed instrument currently existed.

A set of objectives was developed for the test to examine the basic knowledge and understanding of the fundamental commands and concepts within the LogoWriter language. The Logo assessment work of Horton and Ryba (1986) was used to help focus test objectives on six basic levels of Logo programming: 1) basic turtle commands, 2) repeat commands, 3) defining procedures, 4) subprocedures and superprocedures, 5) inputs and variables, and 6) recursion. Specific test objectives for each of these levels were created with reference to the guided instructional text: LogoWorks: Lessons in Logo, by Cory and Walker (1985).

To help verify the content validity of the LogoWriter basic comprehension test, four field experts were asked to evaluate the instrument. These experts included one professor in computer science, one doctoral candidate in computer science, one professor in educational computing, and one professor in instructional psychology. The instructional psychology professor was from the University of

Belgium; all other experts were from Iowa State University. All four experts were very familiar with the Logo language and all but the doctoral candidate had published research in the area. Experts were asked to provide extensive written critiques both for the test objectives and for the test questions. Based on these written critiques, and through verbal discussions, both the objectives and questions were modified to better represent and evaluate basic comprehension of the LogoWriter language. To secure a reliability estimate, the instrument was given to a summer class of Secondary Education 101 students at the conclusion of the LogoWriter unit. The KR-20 reliability estimate was .82 for this sample of 18 students. Upon review of the item analysis from this administration, and through further discussion with some of the experts, additional slight modifications to the response choices of three of the questions were made.

Summary of the research instruments

In summary, three instruments were used in the study to assess far and near transfer effects from the instructional treatment. The Cognitive Ability Test - Nonverbal Battery was used as an operational measure of general analogical reasoning and assumed to evaluate far transfer effects. It was a standardized instrument and has been commonly used by researchers investigating analogical reasoning and its training. The reuse of subprocedures programming test was an instrument which had been modified from the work of Kurland,

Clement, Mawby, and Pea (1987), to focus more directly on student reuse of subprocedures, and was assumed to evaluate near transfer effects of the treatment. This test was an on-line programming activity which incorporated five distinct programming problems that permitted a reuse of subprocedures between them. The LogoWriter basic comprehension test was a locally created and locally standardized multiple choice instrument used to provide an assessment of the basic knowledge and comprehension of the LogoWriter language. It was developed and included in the study to secure evidence that any observed differences in the student reuse of subprocedures were not merely due to differences in lower level comprehension of the LogoWriter language.

Pilot of programming instruments and instructional materials

The programming measurement instruments, and treatment instructional materials, were piloted to ensure their appropriateness for use in the study. A regularly scheduled Secondary Education 101 class, during the summer of 1988, was used for this purpose.

The reuse of subprocedures programming test, used as an on-line instrument, was given as part of the midterm exam in the course. The mean number of subprocedures reused by the students for this pilot of the instrument was 3.38, with a standard deviation of 2.93. This suggested an average use of three to four subprocedures, with about two thirds of the group varying between zero subprocedures reused

and six subprocedures reused, and roughly one third of the group using more than six subprocedures. Such statistics suggested there would be enough variance on the instrument to differentiate effectively between students. In addition, the hour time limit and on-line format of the test seemed to be appropriate and to facilitate student efforts.

The LogoWriter basic comprehension test was also administered during the LogoWriter unit as part of the students midterm exam. Student scores on the 30 question multiple choice test ranged from 12 to 29 correct, with a mean of 21.94, and a standard deviation of 4.36 questions. The KR-20 reliability estimate for the test was .82 for the 18 students, with a standard error of measurement of 1.87. The average student percent score on the test was 76%. The KR-20 reliability estimate and standard error of measurement indicated that the test was fairly reliable for a 30 question multiple choice test, as suggested by Borg and Gall (1983, pp. 281-288), and Stanley and Hopkins (1972, pp. 118-126). Thus, the instrument was considered to be reliable enough for eventual use in the actual study.

The analogical reasoning based Logo instructional materials were also piloted during the regular LogoWriter unit of the course. During the unit, instructional materials were critiqued and revised for both content and sequence by the dissertation author and his major professor. Careful researcher notes, student feedback, and daily video taped classroom observations were used to suggest these revisions.

In summary, to ensure appropriateness for the eventual study, the analogical reasoning based instructional materials and study programming instruments were piloted on a summer session of the same class in which the actual study would take place. Instructional materials were carefully critiqued and revised for sequence and content. Programming instruments, including both the online reuse of subprocedures test and the LogoWriter basic comprehension test, were also piloted by incorporating them as part of the midterm exams for the summer class. The formal statistics gathered, in addition to informal but careful observations, suggested that each of the instruments would be appropriate for use in the actual study.

Administration of study instruments

Each of the three study instruments was used in a post-test only format, and was incorporated as part of the regular course schedule in which the student subjects were enrolled. The reuse of subprocedures programming instrument was used as part of a lab midterm for the students, and the LogoWriter basic comprehension test was used as part of the lecture midterm. The Cognitive Ability Test - Nonverbal Battery was not used for course evaluation, but it was required that students take the test. Students were promised feedback on their relative scores and class rankings.

The Cognitive Ability Test - Nonverbal Battery was used as an outcome measure representing far transfer effects of the instructional

treatment. KR-20 Reliabilities were computed for each subtest of this battery along with a reliability score for the battery, and are given along with standard error of measurement statistics in Table 1. However, only the battery composite score, with a KR-20 reliability of .87, was used in statistical tests. This was because individual subtests have as little as 15 questions, and the current technical manual for form four of this test does not report individual reliabilities for battery subtests, only each of the three batteries (Thorndike & Hagen, 1987b).

Table 1
Cognitive Ability Test Nonverbal Battery KR-20 Reliability Estimates
 Computed for the Current Study

<u>Subtest/Battery</u>	<u>Number of Items</u>	<u>Reliability Index</u>	<u>Standard Error of Measurement</u>
Figure Classification	25	.70	2.17
Figure Analogies	25	.74	1.95
Figure Analysis	15	.79	1.67
<u>Nonverbal Battery</u>	<u>65</u>	<u>.87</u>	<u>3.42</u>

The reuse of subprocedures programming test was used as an outcome measure representing near transfer effects of the instructional treatment. Descriptive statistics for this test indicated

that the overall mean for reuse of subprocedures on the test was 2.97, with an overall standard deviation of 3.09.

The LogoWriter basic comprehension test was administered to determine relative comprehension of the LogoWriter language, and to help provide insight into results from the reuse of subprocedures programming instrument. It consisted of 30 multiple choice questions. Students were given one hour in which to complete the exam, although no student used the complete time allotted. The KR-20 reliability computed for the test was .71, with a standard error of measurement of 2.08 for the raw scores. The average test score was 75%.

Administration of each of the study instruments, as well as use of student subjects in the study, was pre-approved by the Iowa State University Human Subjects committee. Written approval for the study was received on September 9, 1989.

Research Design and Procedures

This study investigated the potential of guided Logo programming instruction for use in the development and transfer of analogical reasoning. In this pursuit, two potential effects were considered: 1) the far transfer effects of instruction, as measured by a standardized test associated with general analogical reasoning, and 2) the near

transfer effects of instruction, as measured by a constructed programming test focused on the reuse of program subprocedures.

Research design

The study used a randomized post-test only control group design in looking at both the far and near transfer effects. To investigate these transfer effects, two dependent variables were used. Student composite score on the Cognitive Ability Test - Nonverbal Battery was assumed to represent the far transfer effects of the instructional treatment, and the student reuse of subprocedure score, on a constructed programming test, was assumed to represent the near transfer effects. Pretests were not given due to the potential interaction of the instruments.

Instructional treatment acted as the independent variable in the study, with subjects randomly assigned to one of two treatment groups: 1) the experimental group, experiencing Logo programming instruction systematically incorporating analogical reasoning training, or 2) the control group, experiencing Logo programming instruction taught in a more traditional approach, not explicitly incorporating analogical reasoning based training. Instructional content was carefully controlled, with an indication of relative comprehension provided by a constructed LogoWriter basic comprehension test. Students were randomly assigned, by individual student, to the respective treatment group by use of a table of random numbers and a class roster.

<u>Group</u>	<u>Week 1</u>	<u>Week 4-6</u>		<u>Week 7</u>		
Experimental	Q	R	X	O ₁	O ₂	O ₃
Control	Q	R	Y	O ₁	O ₂	O ₃

- Q** - Initial questionnaire administered
R - Randomized by student
X - Instructional treatment of guided Logo programming, systematically incorporating analogical reasoning
Y - Instructional treatment of traditional Logo programming
O₁ - Reuse of subprocedures programming test
O₂ - Cognitive Abilities Test - Nonverbal Battery
O₃ - LogoWriter basic comprehension test

FIGURE 1. Sequence of study events

The initial questionnaire to determine sample characteristics was given during week one of the course with the instructional treatments beginning in the fourth week. The instructional treatments continued through week six, with outcome measures administered in the week directly following this period (see Figure 1). During the second and third week of the course, preceding instructional treatments, students received two weeks of instruction on AppleWorks wordprocessing as a regular part of the course schedule. However, students were not randomly divided into the separate treatment groups until the beginning of the Logo programming unit used in the study.

Procedures for additional research controls

Some additional research controls were needed due to the necessity of operating within the parameters of a regularly scheduled university class of large enrollment. The study was conducted as a three week programming unit in the regular course activities of Secondary Education 101. The course was an educational computing class for preservice teachers that had a typical enrollment of about 170 students. This class was structured in two parts: 1) a twice a week one hour lecture format, in which students sat in a large auditorium and participated in large group lecture and demonstration, and 2) a once a week two hour laboratory format, where students worked in groups of approximately 18-20 in an educational computing lab, with each student on a single computer.

The class has traditionally been structured so that students are first introduced to instructional materials in the lecture setting and then required to apply and practice this material within the laboratory settings. During the semester of the study, two lecture sections were held; one late morning lecture section that included students from five laboratory sections, and one early afternoon lecture section, that included students from four laboratory sections. Since treatment was randomly assigned by student, it was necessary to split each lecture section and each laboratory section into two distinct groups, which then went through their respective instructional treatments simultaneously. This necessity required the use of two separate

instructors and two separate rooms for each class meeting during the study.

Instructor controls To control for instructor influence in the lecture sessions, instruction was alternated in each group between the two lecture instructors - the major professor, and the dissertation author. To control for instructor influence in the laboratory sections, instruction was alternated between the regular laboratory instructor and an additional instructor who was either the dissertation author or a senior laboratory instructor. From an instructor's perspective, all instructors taught both instructional treatments, and each was scheduled in either treatment 50% of his or her teaching time.

Pedagogical controls To help ensure that instructors utilized a pure instructional treatment and did not mix instructional techniques, careful content and pedagogical outlines were issued for each class meeting (Appendices G, H, I, and J). These instructional outlines, along with the student instructional sheets issued for each class activity, provided a careful sequence and content of instruction for use by the instructors in both treatment groups. These detailed instructional outlines were used in all lecture and laboratory meetings for the entire unit.

In addition to daily instructional outlines, individualized training for each class session was also administered to help prevent mixing of instructional treatments. For laboratory sessions, this training involved first carefully going over and discussing the methodology for

the class session, and then having the laboratory instructor "micro-teach" the instruction in front of the dissertation author to ensure purity of methodology. Preparations for the lecture sessions were more informal. This preparation consisted of the two lecture instructors meeting prior to the instruction to discuss methodology and content. This approach was deemed appropriate since both lecture instructors had participated in the pilot of the instructional materials. Finally, to help ensure that the training and materials were being correctly implemented by the laboratory instructors, the major professor periodically monitored laboratory instruction in person, and the dissertation author periodically monitored laboratory instruction by use of video tape.

Analogical reasoning introduction control In addition to instructor and pedagogical controls already discussed, a further content control was incorporated to ensure that any differences between the experimental and control groups were not due to just the initial brief introduction of analogical reasoning and the four Sternberg component processes. In order to prevent treatment effects from being too heavily influenced by this single aspect of the instructional treatment, both groups were given the same 30 minute introduction to analogical reasoning and to the component processes of analogical reasoning before the programming unit began. This brief introduction included a basic definition of analogical reasoning, three short examples of analogical reasoning, a brief definition of the theoretical

components of the skill, and a single example of the components used together (see Appendix M for transparencies). Following this brief introduction, students began their respective instructional treatments with the experimental group continuing to incorporate and emphasize the analogical reasoning components in the planning of their programs, and the control group using their own strategies for program planning.

Room controls In addition to instructor and content controls, it was also necessary to provide some control for potential room differences. Since classes were split into two groups meeting simultaneously, two separate rooms were necessary for each class. To provide that the classroom did not enter into treatment effects, schedules were adjusted so that each of the rooms was used equally between experimental and control treatments. Thus, half of the experimental sections, and half of the control sections, were scheduled in each of the available classrooms. Individual experimental and control sections always met in the same location, however, and careful attendance records were kept to prevent students from showing up in the wrong room. In addition, instructional aids, such as overheads, liquid crystal projection devices, etc. were kept consistent throughout all classrooms.

Procedures for administration of study instruments:

Each of the three outcome measures for the study was administered in the week directly following the programming unit. The Cognitive Abilities Test - Nonverbal Battery was administered as a required, but ungraded course activity. Students were told that they would be given feedback on the results of the test; and the importance of a good effort on the test was emphasized. The reuse of subprocedures programming test was used as part of the graded hands-on laboratory midterm for the course. Similarly, the LogoWriter basic comprehension test was used as part of the graded lecture midterm for the course. All instruments were administered by the dissertation author, with both treatment groups taking the test simultaneously in the same testing room to provide consistency and similarity in testing environment.

Directional Hypotheses and Analysis of Data

Since the guided Logo instruction was designed to facilitate the development and transfer of analogical reasoning, two directional hypotheses were used in the study. These hypotheses predicted an improved performance for the experimental group on both of the study instruments representing transfer: 1) the Cognitive Ability Test Nonverbal Battery, representing far transfer of learning, and 2) the

reuse of subprocedures programming test, representing near transfer of learning.

Directional hypotheses

Thus, in looking at the potential transfer effects of Logo programming instruction systematically guided toward analogical reasoning, the following two directional hypotheses were empirically tested:

Hypothesis 1:

Students experiencing programming instruction, explicitly guided toward analogical reasoning development, will have a significantly higher mean composite score on the Cognitive Abilities Test - Nonverbal Battery, than will a control group experiencing programming instruction without explicit guidance.

Hypothesis 2:

Students experiencing programming instruction, explicitly guided toward analogical reasoning development, will demonstrate a significantly higher mean reuse of subprocedures, on a constructed test of programming problems, than will a control group experiencing programming instruction without explicit guidance.

Analysis of data procedures

Statistical procedures focused on testing of the two main hypotheses for the study. These procedures compared mean scores between treatment groups for the Cognitive Ability Test - Nonverbal Battery, and for the reuse of subprocedures constructed programming test. A third outcome measure, the LogoWriter basic comprehension test, was also administered to provide an indication of relative comprehension of the LogoWriter language, and to help interpret any observed differences on the reuse of subprocedures programming instrument. Additional auxiliary procedures were also included to support the investigative nature of the study.

The Cognitive Ability Test The Cognitive Ability Test - Nonverbal Battery was the outcome measure for the first hypothesis, and assumed to represent far transfer effects of the instructional treatment. This test produced four separate scores for the Nonverbal Battery: 1) a Figure Classification subtest score, 2) a Figure Analogies subtest score, 3) a Figure Analysis subtest score, and 4) a composite score of the three subtests. The composite score was used to test the first hypothesis and operated as the dependent variable. Basic treatment differences on this variable were initially analyzed by use of the t-test statistical technique.

The reuse of subprocedures programming test The reuse of subprocedures constructed programming test was the outcome measure for the second hypothesis, and assumed to represent near

transfer effects of the instructional treatment. This test produced a single score that indicated the student reuse of subprocedures between the five programming problems. Raw data were transformed by use of a logarithmic transformation to achieve uniform variance. Similar to the first hypothesis, basic treatment differences on this variable were initially analyzed by use of the t-test statistical technique.

The LogoWriter basic comprehension test The LogoWriter basic comprehension test was administered to verify that the programming instructional content, taught with the two different treatment methodologies, was relatively equally comprehended by both treatment groups. This test was considered particularly important for providing empirical evidence that differences observed in the reuse of subprocedures were not merely the result of lower level differences or inconsistencies in overall comprehension of the LogoWriter language. The statistical technique used with this test was a standard t-test looking at the significance of differences between treatment means.

Auxiliary analyses concerning possible interactions It was considered that additional sources of variation might still be masking, or interacting with, instructional treatment, even after initial randomization procedures. Such a situation is not uncommon in educational studies, and often suggests a factorial design (Borg and Gall, 1983, pp. 685-691). Thus, auxiliary post-hoc analyses were performed using various independent factors and covariates in factorial designs attempting to hold particular sources of variation constant to

determine treatment effects. This was done to help support the investigative nature of the study. Data for these additional independent variables were secured from the questionnaire given at the beginning of the study to look at sample characteristics. Two categorical variables, gender and college year, were used as independent factors in these tests. Two continuous variables, age and a self-reported computer nervousness score, were used as covariates within the factorial designs.

Age was considered an appropriate covariate since it has been consistently shown to be a possible source of variation in cognitive processing, especially processing involving inductive strategies related to analogical reasoning (Alderton, 1985; Bisanz, 1984; Sternberg, 1982; Goldman, 1982). Computer anxiety, represented in the study by the self-reported computer nervousness score, was also considered to be a possible source of variation, and was used as a covariate in post-hoc analyses. This procedure was considered appropriate since many studies have shown that computer anxiety can be a powerful emotional state with effects on both behavior and learning (see Cambre, 1985, for a review).

Gender, operating as a categorical variable, also was considered as a possible source of variation and used as an independent factor in the factorial designs. Significant gender differences have often been reported in the learning of mathematics and computer science concepts, although this trend seems to be changing (Schildkamp-

Kundiger, 1982). College year was also used as a factor in the post-hoc statistical designs. College year, although probably representing a mixture of student characteristics, may give some indication of general crystallized ability in a variety of content areas. Crystallized ability, with its relationship to prior achievement, has been suggested as a source of cognitive treatment interaction (Snow, 1980; Hart, 1986, also see Tobias 1976).

Several available variables were not used as covariates or factors for particular reasons relating to responses of the sample on the initial questionnaire. Grade point average was not used as a factor, or covariate, because college freshmen within the sample did not yet have a college GPA and left this question blank on the questionnaire. Computer experience and math experience were also not used as covariates or factors due to the extreme homogeneity of the sample on these variables. Thus, only the four specific variables of gender, college year, age, and self-reported computer nervousness, were used in auxiliary analyses attempting to further control for the possible interaction of additional independent variables.

Auxiliary analysis of the constructed programming test Since the study was investigative in nature, auxiliary descriptive data relating to student performance on the constructed programming test was also gathered and compared between treatment groups. These data were secured from the further analysis and scoring of student programs, and were used to help suggest further research, and to help interpret

results from the reuse of subprocedures programming instrument. As well as the targeted reuse of subprocedures score, used in testing of the second hypothesis, scores were also computed for four other aspects of student programming performance. These were: 1) the number of programming problems completed successfully by the student, 2) the number of commands used per successful problem by the student, 3) whether the student used variables within the test, and 4) whether the student used recursion within the test.

Auxiliary correlations with the cognitive ability test To provide additional insight into study results, each of the programming variables was correlated with the composite score on the Cognitive Ability Test - Nonverbal Battery. This procedure was done to examine the strength of the relationship between student reuse of subprocedures and general analogical reasoning, as measured by study instruments. The theoretical relationship between reuse of subprocedures and analogical reasoning acted as a premise for the second hypothesis of the study. The non-parametric Spearman rank order correlation technique was used to provide correlations and relative significance levels between variables.

Summary of analysis of data procedures In summary, statistical procedures used in the study focused on the testing of the two main hypotheses for the study. Hypothesis one predicted that the experimental group would achieve a significantly higher mean score on the Cognitive Ability Test - Nonverbal Battery. This hypothesis was

initially tested by use of a t-test on the composite score means. Similarly, hypothesis two predicted a greater reuse of subprocedures on the constructed programming test by the experimental group. This hypothesis was also tested by use of an initial t-test. However, the experimental group's raw scores had a significantly higher variance, and a logarithmic transformation of the data was needed to stabilize variances between groups.

In addition to statistical tests for the two study hypotheses, treatment groups were compared on relative comprehension of the LogoWriter language. To compare treatment groups, means were statistically analyzed for the LogoWriter basic comprehension test by use of a statistical t-test.

To support the investigative nature of the study, auxiliary study analyses were also performed and reported. These included several analysis of variance tests to further investigate study hypotheses by controlling for the influence of additional independent variables. Auxiliary results also included descriptive statistics reported on the reuse of subprocedures programming instrument, to provide insight into results from the second hypothesis. Various correlations were also performed between composite scores on the Cognitive Ability Test - Nonverbal Battery and various scores from the reuse of subprocedures programming instrument. The purpose was to verify the relative strength of the relationship between general analogical reasoning and reuse of subprocedures in the current study.

Summary

In this chapter the methodology of the study was described in five sections: 1) subjects, 2) instructional treatments 3) research instruments, 4) research design and procedures, and 5) directional hypotheses and analysis of data procedures. These sections discussed a methodology supporting the overall purpose of the study - to investigate the potential of guided Logo programming instruction for use in the development and transfer of analogical reasoning. Each of these sections will be briefly summarized.

In the beginning section of this chapter, a description of the subjects was given. By an initial survey, it was found that the majority of the subjects were female elementary education majors. However, a small but significant number of the subjects were male agricultural majors, with a variety of other majors of mixed gender also represented. Also, the sample of subjects generally had few programming and mathematics courses in their backgrounds, with a significant number of the subjects expressing some nervousness regarding their upcoming computer experiences.

In the second section, descriptions of the instructional treatments were given. The experimental treatment consisted of Logo programming instruction incorporating an overall structure to emphasize and train general analogical reasoning. This treatment

relied heavily on the Logo research of Swan and Black (1987), and on the component analogical reasoning research of Sternberg (1977a, b), in the design of its pedagogical approach. The control group treatment consisted of Logo programming instruction taught in a more traditional way, emphasizing student freedom in the planning of solutions to assigned problems. To help ensure that instructional methodology was the only difference between the treatments, programming content was kept the same between the two groups.

In the third section, the three research instruments were described. The Cognitive Ability Test - Nonverbal Battery was used to represent far transfer effects of general analogical reasoning. It was a standardized instrument composed of three subtests. The reuse of subprocedures programming test was used to represent near transfer effects involving student reuse of subprocedures between programming problems. It was an on-line programming instrument modified from the research of Kurland, Clement, Mawby, and Pea (1987). Additionally, a third instrument, the multiple choice LogoWriter basic comprehension test, was created to help secure evidence that any observed differences on the reuse of subprocedures programming test were not merely due to lower level differences in general comprehension of the LogoWriter language.

The third section also described the initial pilot of the instructional materials and programming instruments. These materials and instruments were piloted on a much smaller but similar

group of subjects to ensure their appropriateness for use in the actual study. A careful critique of these materials, as well as minor revisions, were completed at that time. Finally, the descriptive statistics associated with the administration of these instruments to the actual study sample were reported and discussed.

In the fourth section, the general research procedures and research design of the study were presented. The study used a randomized post-test only control group design in looking at both near and far transfer effects of the instructional treatment. Instructional treatment acted as the independent variable. Composite score on the Cognitive Ability Test - Nonverbal Battery, and reuse of subprocedures score on a constructed programming test, acted as the dependent variables for the study. Instructional content was carefully controlled, with an indication of the relative comprehension of that content provided by a constructed LogoWriter basic comprehension test. Various other research controls were also incorporated to help remove the potential effects of instructor and room influences. The measurement instruments for the study, as well as both instructional treatments, were incorporated into the student's general course schedule, with the programming instruments operating as graded activities in the course.

In the last section, section five, the directional hypotheses and statistical procedures were stated for the study. These hypotheses predicted significantly higher mean scores on both dependent

measures, representing far and near transfer effects, for the experimental instruction emphasizing analogical reasoning. To test these hypotheses, a t-test was completed on the sample means of the treatment groups for each of the two dependent measures. Auxiliary analyses were also done to investigate the possible interaction of other independent variables with the instructional treatment. In addition, a t-test was completed on the mean scores for the LogoWriter basic comprehension test; the test provided insight into relative comprehension of the LogoWriter language and assisted in the interpretation of results from the second hypothesis. Finally, various descriptive and correlational statistics were also reported, helping both to support the investigative nature of the study and to verify study assumptions. The results of each of these statistical tests are reported in Chapter Four.

CHAPTER IV: RESULTS

The purpose of this study was to investigate the potential of guided Logo programming instruction for use in the development and transfer of analogical reasoning. To achieve this purpose, the study focused on two possible transfer effects of the guided instruction: 1) the far transfer of learning, as measured by a standardized test associated with general analogical reasoning ability, and 2) the near transfer of learning, as measured by a constructed programming test that looked at the reuse of subprocedures between programming problems.

As discussed in Chapter Three, the study used a post-test only, control group design to look at both near and far transfer effects. Instructional treatment acted as the independent variable in the study. The experimental treatment consisted of programming instruction carefully structured to emphasize general analogical reasoning in the development of student solutions to programming problems. The control treatment, in contrast, consisted of programming instruction taught using a more traditional Logo approach, with students given greater freedom to develop and test their own solution strategies.

Dependent measures used to investigate differences in the effectiveness of these two treatments were the Cognitive Ability Test - Nonverbal Battery, and a constructed programming test looking at the reuse of subprocedures between programming problems. Additionally, a basic LogoWriter comprehension test was constructed and

administered to indicate any treatment differences in relative comprehension of the LogoWriter language, and to help interpret results on the reuse of subprocedures programming instrument.

This chapter is divided into four sections. In the first section, statistical results for the first study hypothesis are reported. These results examined far transfer effects of the instructional treatment. In the second section, statistical results are reported for the second hypothesis, that examined near transfer effects. In the third section, results are reported for the LogoWriter basic comprehension test, used to examine relative comprehension of the instructional content, and to provide insight into results from the second hypothesis. In the fourth section auxiliary results for the study are reported. These results include: 1) statistical procedures controlling for the interaction of independent variables, 2) descriptive statistics examining the reuse of subprocedures test, and 3) correlational data exploring the relationship between reuse of subprocedures and analogical reasoning.

Hypothesis One Results

In examining the potential of guided Logo programming instruction for use in the development and transfer of analogical reasoning, the first hypothesis tested was:

Hypothesis 1

Students experiencing programming instruction, explicitly guided toward analogical reasoning development, will have a significantly higher mean composite score on the Cognitive Abilities Test - Nonverbal Battery, than will a control group experiencing programming instruction without explicit guidance.

This hypothesis used the Cognitive Ability Test - Nonverbal Battery as an outcome measure representing the far transfer effects of instruction. The battery composite score was used as the dependent variable in all statistical tests of this hypothesis, and consisted of a sum of the three battery subtests: 1) Figure Classification, 2) Figure Analogies, and 3) Figure Analysis.

Initial hypothesis test

To test the difference between the means for the composite scores of the experimental and control groups, a standard t-test was performed with the results reported in Table 2. The t-test value of -0.28 indicated that there was no significant difference between treatment means, $p < .361$. An associated F-statistic of 1.10 indicated that the equal variances assumption of the t-test had been met. Thus, initial results for the first hypothesis implied that both the guided Logo instruction, acting as the experimental treatment, and the traditional Logo instruction, acting as the control treatment, had similar effects

on the Cognitive Ability Test - Nonverbal Battery, as indicated by the composite means for this battery.

Table 2: Hypothesis 1
Cognitive Ability Test - Nonverbal Battery Comparison of Mean
Composite Scores for Experimental and Control Treatment Groups

Group	N	Mean	S.D.	t-Value	1-Tailed Probability
Experimental ^a	72	34.97	9.38	-0.28	.361
Control ^b	72	35.42	9.84		

^aLogo systematically guided toward analogical reasoning.

^bLogo taught in a traditional, exploratory approach.

Hypothesis Two Results

In looking at the potential of guided Logo programming instruction for use in the development and transfer of analogical reasoning, the second hypothesis tested was:

Hypothesis 2

Students experiencing programming instruction, explicitly guided toward analogical reasoning, will demonstrate a significantly higher mean reuse of subprocedures, on a constructed test of

programming problems, than will a control group experiencing programming instruction without explicit guidance.

This hypothesis test used a reuse of subprocedures score, computed by the steps discussed in Chapter Three, as an outcome measure representing the near transfer effects of instructional treatment. Descriptive statistics for this test are given in Table 3.

Table 3: Hypothesis 2
Reuse of Subprocedures Programming Test Descriptive Statistics for
the Reuse of Subprocedures Raw Score for Both Treatment Groups

Treatment Group	N	Mean	Standard Deviation	Variance
Experimental ^a	72	3.28	3.43	11.78
Control ^b	72	2.69	2.69	7.26

^aLogo systematically guided toward analogical reasoning.

^bLogo taught in a traditional, exploratory approach.

These statistics indicated that the guided Logo group, acting as the experimental instruction, had a larger variance in raw scores than did the traditionally instructed Logo group, with variances of 11.78 and 7.26 respectively. A Hartley's test for homogeneity of variances found an F statistic of 1.62, and confirmed that this difference in variances was significant at $p < .043$. Since the mean of each group was

approximately equal to its corresponding standard deviation, a basic logarithmic transformation of the data was performed, as suggested by Ott (1984, p. 341-342), to stabilize variances. Statistical tests were then completed on this transformed data.

Initial Hypothesis Test

To test the means of the transformed reuse of subprocedure scores, a standard t-test was performed with the results reported in Table 4.

Table 4: Hypothesis 2
Reuse of Subprocedures Programming Test Comparison of Means for
 the Transformed Reuse of Subprocedures Scores for
 Both Treatment Groups

Group	N	Mean ^a	S.D.	t-Value	1-Tailed Probability
Experimental ^b	72	.436	.358	-0.45	.327
Control ^c	72	.465	.404		

^aRaw data transformed logarithmically to achieve uniform variance.

^bLogo systematically guided toward analogical reasoning.

^cLogo taught in a traditional, exploratory approach.

The t-test value of -0.45 indicated that there was no significant difference between the treatment means of the transformed scores at $p < .327$. An associated F statistic of 1.27 suggested that the equal

variance assumption was now met. Thus, initial results for hypothesis two showed no significant difference in the means for the transformed reuse of subprocedure scores, implying that both instructional treatments had similar effects on the mean score for student reuse of subprocedures.

Results for the LogoWriter Basic Comprehension Test

The LogoWriter basic comprehension test was developed to examine the relative comprehension of basic commands and concepts in the LogoWriter programming language, operating as the instructional content for the study. Such a test verifying relative comprehension of the instructional content was important, since this study sought to contrast the effects of two different instructional methodologies when teaching the same instructional content. This test also provided possible insight into the results of the second hypothesis, since any observed differences in basic comprehension of the LogoWriter language would directly impact results dealing with the higher level programming construct of reuse of subprocedures. To test the difference between the experimental and control group means for scores on the LogoWriter basic comprehension test, a standard t-test was performed with the results reported in Table 5. The t-test value of -0.83 indicated that there was no significant difference between treatment means, with $p < .408$. An associated F-

statistic of 1.29 indicated that the equal variances assumption of the t-test had been met. Thus, results for the LogoWriter basic comprehension test indicated that both treatment groups had achieved a statistically equal understanding of the basic commands and concepts in the LogoWriter language, as suggested by mean scores on the test.

**Table 5: Instructional Content Comprehension
LogoWriter Basic Comprehension Test Comparison of Means Scores
for Both Experimental and Control Treatment Groups**

Group	N	Mean	S.D.	t-Value	2-Tailed Probability
Experimental ^a	71	23.18	3.95	-0.83	.408
Control ^b	72	22.67	3.48		

^aLogo systematically guided toward analogical reasoning.

^bLogo taught in a traditional, exploratory approach.

Auxiliary Results

Since this study was investigative in nature, various auxiliary analyses were incorporated to help provide further insight into study results. Auxiliary analyses included several analysis of variance statistical tests, associated with hypothesis one and two, that further controlled for additional independent variables. Auxiliary procedures also included various descriptive statistics looking more closely at the

reuse of subprocedures programming test. Finally, several correlations were also performed, to investigate the relationship between reuse of subprocedures and analogical reasoning, as measured by study instruments.

Auxiliary analyses for hypothesis one

It was considered that additional sources of variation, from other independent variables associated with individual student characteristics, might be interacting with instructional treatment to mask transfer effects. Such a situation is common in educational studies, and often suggests a factorial design (Borg and Gall, 1983, pp. 685 - 691). As discussed in Chapter Three, two categorical variables, gender and college year, were used as additional independent factors in further analysis of variance tests using a factorial design. Although randomization should have equated groups with respect to age and computer nervousness, these continuous variables were also entered as covariates in the factorial designs to be sure this source of variation was statistically removed.

Treatment with covariates For the first auxiliary analysis, the variables of age and self-reported computer nervousness were added as covariates in an analysis of variance statistical test of the treatment composite means (Table 6, p. 132). Both age and self-reported computer nervousness were found to be significant covariates, with

$p < .002$ and $p < .001$ respectively; however, the main effects of instructional treatment were still not significant, with $p < .702$. Thus, after controlling for age and initial computer nervousness, there still was no significant difference between the means of the composite scores, for the guided Logo and traditional Logo instructional groups, on the Cognitive Ability Test - Nonverbal Battery.

Treatment by gender with covariates The second auxiliary analysis included the independent factor of gender along with the factor of instructional treatment, for a 2 by 2 factorial design still incorporating the covariates of age and computer nervousness (Table 7, p. 133). No significant differences were found for either gender, $p < .487$, or gender/treatment interaction, $p < .661$, on the means of the composite scores for the Cognitive Ability Test - Nonverbal Battery. Age and computer nervousness were still significant as covariates with $p < .002$, and $p < .001$ respectively.

Treatment by college year with covariates The third and last auxiliary analysis for hypothesis one used college year as an independent factor using a 2 by 4 factorial design (Table 8, p. 134). Computer nervousness was included as covariate and found to be significant at $p < .002$. Age was not included as a covariate due to the strong correlation between age and college year in this study. Although no significant difference was found for college year alone, $p < .209$, a significant difference was found for college year and instructional treatment interaction, $p < .034$. Descriptive statistics

indicated a pattern for the interaction: freshmen achieved a higher mean score with the guided Logo instruction; sophomores performed relatively equally between instructional treatments; and juniors and seniors achieved a higher mean score within the traditional Logo instruction. Individual analysis of variance tests were then run for each of the year subgroups, with age and computer nervousness still operating as covariates (Tables 9 - 12, pp. 135 - 138). These tests indicated that the mean for experimental freshmen was significantly higher than the mean for control freshmen, $p < .047$, and the mean for experimental juniors was significantly lower than for control juniors, $p < .029$. No individual significance was found for the sophomore and senior subgroups. Thus, guided Logo instruction produced a statistically higher mean composite score for freshmen, and traditional Logo instruction produced a statistically higher mean composite score for juniors, implying that these two subgroups were responsible for much of the interaction effect between treatment and college year found in the full sample analysis of variance test.

Summary of auxiliary analysis for hypothesis one An initial t-test of treatment means for the composite scores on the Cognitive Ability Test - Nonverbal Battery had found no significant difference between instructional groups. Various auxiliary analysis of variance tests were then performed. These tests used a factorial design incorporating college year and gender as independent factors, and age and self-reported computer nervousness as covariates. Both age and self-

reported computer nervousness were found to be significant covariates. The factor of gender, and its associated interaction with instructional treatment, were not found to be significant sources of variation. However, although a subject's year in college alone was not found to be a significant source of variation, the interaction of instructional treatment and year in college was found to be significant, with $p < .034$. Descriptive statistics indicated a pattern for this interaction with freshmen achieving a higher mean score in the experimental group, sophomores achieving approximately equal scores in both groups, and juniors and seniors achieving higher mean scores in the control group. Individual analysis of variance tests for each college year indicated that the differences between freshmen subgroups, and the differences between junior subgroups, were responsible for much of the interaction effect between college year and treatment.

Auxiliary analyses for hypothesis two

As in hypothesis one, it was considered that additional sources of variation might be interacting with instructional treatment to mask treatment effects. Again, data from the initial sample questionnaire was used to provide additional independent variables for auxiliary analysis of variance tests. The categorical variables of gender and college year were used as independent factors in the factorial designs, with the continuous variables of age and self-reported computer nervousness controlled as covariates.

Treatment with covariates For the first auxiliary analysis of hypothesis two, age and self-reported computer nervousness were added as covariates in an analysis of variance test of means for the transformed reuse of subprocedures scores (Table 13, p. 139). In contrast to the findings of hypothesis one, age and computer nervousness were not found to be significant covariates for hypothesis two, with $p < .793$, and $p < .267$ respectively. No significant difference was found for the means of the transformed reuse of subprocedures score, with $p < .759$, implying no difference in treatment effects on the mean reuse of subprocedures.

Treatment by gender with covariates The second auxiliary analysis for hypothesis two included the independent factor of gender along with instructional treatment to provide a 2 by 2 factorial design (Table 14, p. 140). Age and computer nervousness were again entered as covariates and found to be non-significant, with $p < .792$, and $p < .263$, respectively. Although gender/treatment interaction was found to be non-significant at $p < .364$, gender as a main effect approached but did not achieve significance at $p < .074$. Descriptive statistics indicated that females, with a mean for the transformed scores of .48, had performed slightly better than males with a mean for the transformed scores of .36. A slightly greater contrast was found within the males as a group. Although females performed equally well in each instructional treatment, with similar means of .48, males reused more subprocedures in the experimental group, with a mean of .43

compared to .30. However, when a separate analysis of variance was performed for the males subgroup, an F value of .73 indicated that the mean for experimental males was not significantly larger than the mean for control males, $p < .199$ (Table 15, p. 141). Thus, as with the first auxiliary test for hypothesis two, differences in treatment effects on the mean reuse of subprocedures, this time incorporating gender as an additional factor, were not significant.

Treatment by college year with covariates The third and last auxiliary analysis for hypothesis two used college year as an independent factor with instructional treatment for a 2 by 4 factorial design (Table 16, p. 142). Computer nervousness was included as covariate, but continued to be non-significant at $p < .313$. Age was not entered as a covariate due to its high correlation with college year. No significant difference was found either for college year alone, $p < .995$, or for interaction between college year and treatment, at $p < .986$. Thus, as with the other statistical tests of hypothesis two, differences in treatment effects on the mean reuse of subprocedures could not be considered significant.

Summary of auxiliary analysis for hypothesis two

An initial t-test performed on means for the transformed reuse of subprocedures score indicated no significant difference between treatment means. Several analysis of variance statistical tests were then performed to statistically control for additional independent

variables. Both age and computer nervousness were found to be non-significant covariates within these statistical tests. Although an effect for the gender factor approached significance in these auxiliary tests, overall results still indicated that there was no significant difference in the treatment means of the transformed scores. No significant interaction was also found for student year in college. Thus, further auxiliary analyses, as well as the initial t-test, implied that there was no significant difference in the mean reuse of subprocedures between treatment groups.

Further analysis of the programming test

Additional descriptive statistics related to performance on the constructed reuse of subprocedures programming test were also included in the study results. These statistics were used to clarify results associated with hypothesis two, and to help suggest further research. These descriptive statistics were gathered from the reuse of subprocedures programming test, and described group performance related to several programming aspects: 1) the number of programming problems completed successfully, 2) the number of commands used per successful problem, 3) the use of variables, and 4) the use of recursion.

These additional descriptive statistics from the reuse of subprocedures programming instrument were included in the study for two basic reasons: 1) to aid in discussion of the reuse of

subprocedure results by describing related aspects of student programming performance on that instrument, 2) to help provide suggestions for further research.

Success on particular problems The percent of both treatment groups successfully programming a solution to each of the five programming problems on the reuse of subprocedures programming test is given in Table 17 (p. 143). For each of the problems, the experimental group had a slightly higher percentage of subjects successfully program a solution. Problems were checked by execution of the coded program, with student output considered correct only if it perfectly matched desired output.

The greatest difference between the groups occurred with problem number four, with 13.1% more of the experimental group successfully programming a solution to this problem of similar rectangles. The next highest difference between the groups rested with problem number one, a problem using horizontally positioned rectangles. Problem number three, with shaded rectangles positioned diagonally upward to the right, had a difference between groups of 8.1%. Differences for the other two problems, both incorporating squares, were less, but still in favor of the experimental group, with problem number two and three having a difference of 3.0% and 6.3% respectively. Thus, in summary, the experimental group had a greater percentage of members successfully program a solution to each of the five problems, with the greatest differences in group percent found on

the three problems that incorporated rectangles. The two problems that used squares for the output had relatively less of a difference in group percentages.

Number of commands used on particular problems The number of commands used for each successfully programmed problem was computed, with group means for each problem given in Table 18 (p. 144). To count the number of commands used, a counting process discussed by Kurland, Clement, Mawby, and Pea (1987), was used. In this procedure, three specific counting rules are followed: 1) each Logo primitive is one command, 2) each repeat statement is one command, with repeated commands in the statement counted once, and 3) each procedure call is one command, with commands in the procedure counted only on the initial call.

Treatment groups were relatively close in the mean number of commands that they used for each of the first three problems. Though the experimental group had a slightly lower mean number of commands in each of these problems, this difference was less than a single command. In problems four and five, however, the difference between the experimental and control groups was more substantial. In problem four, the similar rectangle problem, students in the experimental group used an average of 2.2 commands less than the control group in creating their successful programs. This difference was even larger in problem five, incorporating the shaded and diagonally positioned rectangles, with students in the experimental

group using an average of 4.4 commands less than the control group in building their programs. Thus, descriptive statistics suggested little difference in the number of commands used by each group until the later two problems, in which the experimental group used a mean number of commands that was 2.2 and 4.4 commands less than the mean number of commands for the control group.

Percent of group using variables and recursion The reuse of subprocedures tests were further analyzed to determine the percentage of each treatment group using variables and recursion within at least one of the five problems. These percentages are given in Table 19 (p. 145). The group percentages are slightly larger for the experimental treatment in both the use of variables, and the use of recursion. However, these differences are relatively small, with a group difference of 4.2% with variables, and only 2.8% with recursion. Thus, descriptive statistics indicated that the experimental group had a slightly larger percentage of students choosing to use variables and recursion in their programs than the control group.

Summary of programming descriptive statistics The further analysis of the reuse of subprocedures instrument produced additional descriptive statistics dealing with four particular aspects of group performance on this test. The first set of statistics indicated the percent of the sample in each treatment group who had produced successful programs for each of the five test problems. These statistics indicated that the experimental group had a higher percentage of

students who had programmed successfully for each of the five problems, with the largest difference found in the three problems using rectangles. The second set of statistics dealt with the mean number of commands used in successful programs for each group on a specific problem. These statistics indicated that the mean number of commands were virtually the same for each group on the first three problems, but relatively different on the last two problems, with the experimental group using two to four commands less. The third and fourth sets of statistics indicated the percentage of each group choosing to use variables and recursion in at least one problem on the test. These statistics indicated a slightly greater group percentage for the experimental treatment in both the use of variables, and the use of recursion in test problems.

Correlations with the Cognitive Ability Test

Auxiliary results for the study also included various statistical correlations concerned with verification of the statistical relationship between reuse of subprocedures and analogical reasoning, that acted as a research premise for the study. Reuse of subprocedures was targeted as an outcome variable because of the inherent use of analogical reasoning in the purposeful reuse of subprocedures between different programming problems, and because of correlational results supporting this relationship in the work of Clement, Kurland, Mawby, and Pea (1986). Since this inherent relationship acted as a research

premise for the second hypothesis of this study, some further analysis was deemed appropriate to verify the strength of this relationship in the current study. Thus, the reuse of subprocedures score from the constructed programming test was correlated with the composite score on the Cognitive Ability Test - Nonverbal Battery. The purpose was to determine the strength of the relationship between these variables in the current study, as measured by study instruments. Correlations for three other programming variables were performed as well, to provide a relative comparison to other available aspects of programming performance.

The non-parametric Spearman rank order technique was used to perform correlations between composite scores on the Cognitive Ability Test - Nonverbal Battery, and the four programming variables associated with student performance on the complete programming test. The four programming variables considered consisted of scores representing: 1) the reuse of subprocedures between programming problems, 2) the number of programming problems solved successfully, 3) whether the student used variables within the test, and 4) whether the student used recursion within the test. Two other programming variables were not correlated: mean number of commands used per test problem, and percent of the group getting a test problem correct. These two variables were not used as correlational variables because they were associated with group

performance on each individual test problem, rather than with student performance on the complete test.

Results showed that all four of the programming variables considered were correlated to composite score on the Cognitive Ability Test - Nonverbal Battery (Table 20, p. 146). Specifically, number of correct programs correlated at .414, $p < .001$, with reuse of subprocedures at .263, $p < .001$, use of recursion at .225, $p < .003$, and finally, use of variables at .200, $p < .008$. Thus, in this study a significant correlation was found between student reuse of subprocedures and general analogical reasoning ability, as represented by study instruments.

Summary of Study Results

In this chapter, results were presented from an investigation of the potential of guided Logo programming instruction for use in development and transfer of analogical reasoning. Four sections were used to report these results: 1) the results for hypothesis one of the study, 2) the results for hypothesis two of the study, 3) results of the LogoWriter basic comprehension test, looking at relative comprehension of instructional content, and 4) auxiliary investigative results. Auxiliary investigative results included further analysis of variance statistical procedures, descriptive statistics from the reuse of

subprocedures test, and correlations investigating the relationship between reuse of subprocedures and analogical reasoning.

In section one, results were reported for hypothesis one of the study. This hypothesis predicted a higher mean score on the Cognitive Ability Test - Nonverbal Battery for the guided Logo instruction, acting as the experimental group, than for traditional Logo instruction, acting as the control group. Initial testing of hypothesis one by use of a standard t-test found no significant difference in group means.

In section two, results for the second hypothesis of the study were reported. This hypothesis predicted a higher mean reuse of subprocedures on the constructed programming test for the experimental group than for the control group. Group variances were found to be significantly different, with the experimental group having a statistically greater variance than the control group. Raw data were transformed by use of a logarithmic function to achieve uniform variance. Initial testing of the means for the transformed scores, by use of a standard t-test, found no significant difference in group means.

In section three, results were given for the LogoWriter basic comprehension test. This test was used to examine general comprehension of the various commands and concepts in the LogoWriter language that acted as instructional content for the study. A standard t-test indicated that the means for both groups on this instrument were not statistically different. This result suggested that

both treatment groups had achieved a statistically equal understanding of the LogoWriter language, as indicated by the test means.

Finally, in section four, further auxiliary investigative results were reported for the study. These results included analysis of variance procedures, associated with hypothesis one and two, that attempted to control for additional independent variables. In further support of the investigative nature of the study, this section also contained descriptive statistics from the reuse of subprocedures test, and various correlations looking at the relationship between analogical reasoning and reuse of subprocedures.

The auxiliary analysis of hypothesis one, using the analysis of variance statistical technique, found a significant interaction for college year and instructional treatment. Descriptive statistics showed a pattern for this interaction, with freshman performing better in the experimental group, sophomores performing about the same in both groups, and juniors and seniors performing better in the control group. Individual analysis of variance tests, looking at each year separately, indicated that only the subgroups of freshmen and juniors were statistically significant.

Auxiliary analyses for hypothesis two, attempting to control for additional independent variables, found no significant differences. An effect for gender did approach significance, however, with females having a slightly higher mean reuse of subprocedures than males. Also within the male subgroup itself, the experimental treatment achieved a

slightly higher mean than the control treatment. However, this difference was also not statistically significant.

Section four auxiliary results also included additional descriptive statistics from the reuse of subprocedures programming instrument. These statistics were used to provide insight into results from the second hypothesis, and to help suggest further research. The descriptive statistics suggested a slightly better group performance by the experimental group on each of the investigated programming aspects, with the greatest differences between groups occurring on the most difficult problems.

Finally, section four also included correlations of programming scores from the reuse of subprocedures programming instrument with composite scores on the Cognitive Ability Test - Nonverbal Battery. These correlations were included to help verify the relationship between reuse of subprocedures and analogical reasoning, as represented by study instruments. Significant correlations were found between composite score on the Cognitive Ability Test - Nonverbal Battery, and associated programming scores, including the targeted reuse of subprocedures score.

This chapter presented results of a study seeking to investigate the potential of guided Logo programming instruction for use in the development and transfer of analogical reasoning. These results, and the particular insights into this potential that they suggest, are discussed in Chapter Five.

TABLES FOR AUXILIARY RESULTS

Table 6: Auxiliary Results Hypothesis 1
Cognitive Ability Test - Nonverbal Battery Comparison of Mean
Composite Score for Both Treatment Groups with Age and Computer
Nervousness Controlled as Covariates

A. Means and Counts					
Treatment	Experimental^a		Control^b		
	35.42	(72)^c	34.97	(72)	
B. Analysis of Variance					
Source of Variation	Sum of Squares	DF	Mean Square	F	Signif.^d of F
Covariates	1699.83	2	849.91	10.43	0.001
Age	845.20	1	845.20	10.37	.002**
Comp. Nerv.	983.17	1	983.17	12.01	.001***
Main Effects					
Treatment	12.00	1	12.00	0.15	.702
Explained	1711.82	3	570.61	7.00	.001
Residual	11406.73	140	81.48		
Total	13118.56	143	91.74		

^aLogo systematically guided toward analogical reasoning.

^bLogo taught in a traditional, exploratory approach.

^cThe numbers in parentheses denote sample size.

^dThe*** two-tailed significance at the .001 level. The ** indicates two-tailed significance at the .01 level.

Table 7: Auxiliary Results Hypothesis 1
Cognitive Ability Test - Nonverbal Battery Comparison of Mean Composite Score for Both Treatment Groups by Gender, with Age and Computer Nervousness Controlled as Covariates

<u>A. Means and Counts</u>					
Treatment	Experimental ^a		Control ^b		
	35.42 (72) ^c		34.97 (72)		
Gender	Female		Male		
	34.88 (108)		36.14 (36)		
Treatment By Gender	Exp. Fem.	Exp. Male	Cont. Fem.	Cont. Male	
	35.39 (54)	35.50 (18)	34.37 (54)	36.78 (18)	
<u>B. Analysis of Variance</u>					
Source of Variation	Sum of Squares	DF	Mean Square	F	Signif. ^d of F
Covariates	1699.83	2	849.91	10.33	.001
Age	845.20	1	845.20	10.28	.002**
Comp. Nerv.	983.17	1	983.17	11.95	.001***
Main Effects	51.96	2	25.98	0.32	.730
Treatment	11.89	1	11.89	0.14	.705
Gender	39.97	1	39.97	0.49	.487
Interaction					
Treat by Gender	15.92	1	15.92	0.19	.661
Explained	1767.71	5	353.54	4.30	.001
Residual	11350.85	138	82.25		
Total	13118.56	143	91.74		

^aLogo systematically guided toward analogical reasoning.

^bLogo instruction taught in a traditional, exploratory approach.

^cThe numbers in parentheses denote sample size.

^dThe *** denotes two-tailed significance at the .001 level. The ** denotes two-tailed significance at the .01 level.

Table 8: Auxiliary Results Hypothesis 1
Cognitive Ability Test - Nonverbal Battery Comparison of Mean
 Composite Score for Both Treatment Groups by Year in College, with
 Computer Nervousness Controlled as a Covariate

<u>A. Means and Counts</u>					
Treatment	Experimental ^a		Control ^b		
	35.42	(72) ^c	35.07 (71)		
Year in College	Freshmen	Sophomore	Junior	Senior	
	36.14 (44)	36.48 (31)	32.54 (37)	35.97 (31)	
Treatment By Year	Exp. Fresh.	Exp. Soph.	Exp. Junior	Exp. Senior	
	38.95 (22)	36.40 (20)	27.67 (12)	35.17 (18)	
	Cont. Fresh.	Cont. Soph.	Cont. Junior	Cont. Senior	
	33.32 (22)	36.64 (11)	34.88 (25)	37.08 (13)	
<u>B. Analysis of Variance</u>					
Source of Variation	Sum of Squares	DF	Mean Square	F	Signif. ^d of F
Covariates	832.51	1	832.51	10.04	.002
Comp. Nerv.	832.51	1	832.51	10.04	.002**
Main Effects	381.77	4	95.44	1.51	.336
Treatment	24.30	1	24.30	.29	.589
Year in College	380.91	3	126.97	1.53	.209
Interaction					
Treat by Year	738.84	3	246.28	2.97	.034*
Explained	1953.12	8	244.14	2.94	.005
Residual	11113.31	134	82.94		
Total	13066.43	142	92.02		

^aLogo systematically guided toward analogical reasoning.

^bLogo taught in a traditional, exploratory approach.

^cThe numbers in parentheses denote sample size.

^dThe * denotes two-tailed significance at the .05 level. The ** denotes two-tailed significance at the .01 level.

Table 9: Auxiliary Results Hypothesis 1
Cognitive Ability Test - Nonverbal Battery Comparison of Mean
 Composite Score for Freshmen in Both Treatment Groups with Age
 and Computer Nervousness Controlled as Covariates

<u>A. Means and Counts</u>					
Treatment	Experimental ^a		Control ^b		
	38.95 (22) ^c		33.32 (22)		
<u>B. Analysis of Variance</u>					
Source of Variation	Sum of Squares	DF	Mean Square	F	Signif. ^d of F
Covariates	187.19	2	93.60	1.06	.357
Age	142.91	1	142.91	1.61	.211
Comp. Nerv.	16.22	1	16.22	0.18	.671
Main Effects					
Treatment	261.30	1	261.30	2.95	.047*
Explained	448.49	3	149.50	1.69	.185
Residual	3540.69	40	88.52		
Total	3989.18	43	92.77		

^aLogo systematically guided toward analogical reasoning.

^bLogo taught in a traditional, exploratory approach.

^cThe numbers in parentheses denote sample size.

^dThe * indicates one-tailed significance at the .05 level.

**Table 10: Auxiliary Results Hypothesis 1
Cognitive Ability Test - Nonverbal Battery Comparison of Mean
Composite Score for Sophomores in Both Treatment Groups with Age
and Computer Nervousness Controlled as Covariates**

A. Means and Counts					
Treatment	Experimental^a		Control^b		
	36.40	(20)^c		36.64	(11)
B. Analysis of Variance					
Source of Variation	Sum of Squares	DF	Mean Square	F	Signif.^d of F
Covariates	1060.08	2	530.04	8.64	.001
Age	320.06	1	320.06	5.22	.030*
Comp. Nerv.	606.80	1	606.80	9.89	.004**
Main Effects					
Treatment	49.26	1	49.26	0.80	.378
Explained	1109.34	3	369.78	6.03	.003
Residual	1656.40	27	61.35		
Total	2765.74	30	92.19		

^aLogo systematically guided toward analogical reasoning.

^bLogo taught in a traditional, exploratory approach.

^cThe numbers in parentheses denote sample size.

^dThe * indicates two-tailed significance at the .05 level. The ** indicates two-tailed significance at the .01 level.

Table 11: Auxiliary Results Hypothesis 1
Cognitive Ability Test - Nonverbal Battery Comparison of Mean
Composite Score for Juniors in Both Treatment Groups with Age and
Computer Nervousness Controlled as Covariates

<u>A. Means and Counts</u>					
Treatment	Experimental ^a		Control ^b		
	27.67	(12) ^c	34.88	(25)	
<u>B. Analysis of Variance</u>					
Source of Variation	Sum of Squares	DF	Mean Square	F	Signif. ^d of F
Covariates	213.89	2	106.94	1.13	.336
Age	213.88	1	213.88	2.26	.143
Comp. Nerv.	1.31	1	1.31	0.01	.907
Main Effects					
Treatment	366.82	1	366.82	3.87	.029*
Explained	580.71	3	193.57	2.04	.127
Residual	3128.48	33	94.80		
Total	3709.19	36	103.03		

^aLogo systematically guided toward analogical reasoning.

^bLogo taught in a traditional, exploratory approach.

^cThe numbers in parentheses denote sample size.

^dThe * indicates one-tailed significance at the .05 level.

Table 12: Auxiliary Results Hypothesis 1
Cognitive Ability Test - Nonverbal Battery Comparison of Mean
 Composite Score for Seniors in Both Treatment Groups with Age and
 Computer Nervousness Controlled as Covariates

<u>A. Means and Counts</u>					
Treatment	Experimental ^a		Control ^b		
	35.17	(18) ^c	37.08	(13)	
<u>B. Analysis of Variance</u>					
Source of Variation	Sum of Squares	DF	Mean Square	F	Signif. ^d of F
Covariates	668.72	2	334.36	6.16	.006
Age	288.00	1	288.00	5.31	.029*
Comp. Nerv.	492.53	1	492.53	9.08	.006**
Main Effects					
Treatment	99.50	1	99.50	1.83	.187
Explained	768.22	3	256.07	4.72	.009
Residual	1464.75	27	54.25		
Total	2232.97	30	74.43		

^aLogo systematically guided toward analogical reasoning.

^bLogo taught in a traditional, exploratory approach.

^cThe numbers in parentheses denote sample size.

^dThe * indicates two-tailed significance at the .05 level. The ** indicates one-tailed significance at the .01 level.

Table 13: Auxiliary Results Hypothesis 2
Reuse of Subprocedures Programming Test Comparison of Means for
 the Transformed Reuse of Subprocedures Scores for Both Treatment
 Groups, with Age and Computer Nervousness Entered as Covariates

<u>A. Means and Counts^a</u>					
Treatment	Experimental ^b		Control ^c		
	.46 (72) ^d		.44 (72)		
<u>B. Analysis of Variance</u>					
Source of Variation	Sum of Squares	DF	Mean Square	F	Signif. of F
Covariates	.187	2	.094	.637	.530
Age	.010	1	.010	.069	.793
Comp. Nerv.	.182	1	.182	1.243	.267
Main Effects					
Treatment	.014	1	.014	.095	.759
Explained	.201	3	.067	.456	.713
Residual	20.555	140	.147		
Total	20.756	143	.145		

^aRaw data transformed logarithmically to achieve uniform variance.

^bLogo systematically guided toward analogical reasoning.

^cLogo taught in a traditional, exploratory approach.

^dThe numbers in parentheses denote sample size.

Table 14: Auxiliary Results Hypothesis 2
Reuse of Subprocedures Programming Test Comparison of Means for
 the Transformed Reuse of Subprocedures Scores for Both Treatment
 Groups by Gender, with Age and Computer Nervousness Entered as
 Covariates

<u>A. Means and Counts</u>					
Treatment	Experimental ^a		Control ^b		
	0.46 (72) ^c		0.44 (72)		
Gender	Female		Male		
	0.48 (108)		0.36 (36)		
Treatment By Gender	Exp. Fem.	Exp. Male	Cont. Fem.	Cont. Male	
	0.48 (54)	0.43 (18)	0.48 (54)	0.30 (18)	
<u>B. Analysis of Variance</u>					
Source of Variation	Sum of Squares	DF	Mean Square	F	Signif. of F
Covariates	.187	2	.094	.647	.525
Age	.010	1	.010	.070	.792
Comp. Nerv.	.182	1	.182	1.261	.263
Main Effects	.482	2	.241	1.665	.193
Treatment	.013	1	.013	.093	.761
Gender	.468	1	.468	3.233	.074
Interaction					
Treat by Gender	.120	1	.120	.830	.364
Explained	.789	5	.158	1.090	.368
Residual	19.967	138	.145		
Total	20.756	143	.145		

^aLogo systematically guided toward analogical reasoning.

^bLogo taught in a traditional, exploratory approach.

^cThe numbers in parentheses denote sample size.

Table 15: Auxiliary Results Hypothesis 2
Reuse of Subprocedures Programming Test Comparison of Means for
the Transformed Reuse of Subprocedures Scores for Males in Both
Treatment Groups, with Age and Computer Nervousness
Entered as Covariates

A. Means and Counts^a					
Treatment	Experimental^b		Control^c		
	.43 (18)^d		.30 (18)		
B. Analysis of Variance					
Source of Variation	Sum of Squares	DF	Mean Square	F	Signif. of F
Covariates	.135	2	.068	.409	.668
Age	.045	1	.045	.273	.605
Comp. Nerv.	.088	1	.088	.534	.470
Main Effects					
Treatment	.121	1	.121	.732	.199
Explained	.256	3	.085	.517	.674
Residual	5.286	32	.165		
Total	5.542	35	.158		

^aRaw data transformed logarithmically to achieve uniform variance.

^bLogo systematically guided toward analogical reasoning.

^cLogo taught in a traditional, exploratory approach.

^dThe numbers in parentheses denote sample size.

Table 16: Auxiliary Results Hypothesis 2
Reuse of Subprocedures Programming Test Comparison of Means for
 the Transformed Reuse of Subprocedures Scores for Both Treatment
 Groups by College Year, with Computer Nervousness Entered
 as a Covariate

<u>A. Means and Counts^a</u>					
Treatment	Experimental ^b		Control ^c		
	.46 (72) ^d		.44 (71)		
Year in College	Freshmen	Sophomore	Junior	Senior	
	.45 (44)	.44 (31)	.45 (37)	.48 (31)	
Treatment By Year	Exp. Fresh.	Exp. Soph.	Exp. Junior	Exp. Senior	
	.46 (22)	.45 (20)	.45 (12)	.50 (18)	
	Cont. Fresh.	Cont. Soph.	Cont. Junior	Cont. Senior	
	.44 (22)	.43 (11)	.45 (25)	.45 (13)	
<u>B. Analysis of Variance</u>					
Source of Variation	Sum of Squares	DF	Mean Square	F	Signif. of F
Covariates	.156	1	.156	1.024	.313
Comp. Nerv.	.156	1	.156	1.024	.313
Main Effects	.020	4	.005	.033	.998
Treatment	.009	1	.009	.062	.803
Year in College	.011	3	.004	.024	.995
Interaction					
Treat by Year	.022	3	.007	.048	.986
Explained	.198	8	.025	.163	.995
Residual	20.354	134	.152		
Total	20.552	142	.145		

^aRaw data transformed logarithmically to achieve uniform variance.

^bLogo instruction systematically guided toward analogical reasoning.

^cLogo instruction taught in a traditional, exploratory approach.

^dThe numbers in parentheses denote sample size.

Table 17: Auxiliary Results
Programming Instrument Descriptive Statistics Percent of Treatment
 Group Getting Specific Problems Correct on the Reuse of
 Subprocedures Programming Instrument

Test Problems:	Prob. 1	Prob. 2	Prob. 3	Prob. 4	Prob.5
Experimental^a	84.7%	76.4%	48.6%	61.1%	22.2%
Control^b	74.7%	73.4%	42.3%	47.9%	14.1%

^aLogo systematically guided toward analogical reasoning.

^bLogo taught in a traditional, exploratory approach.

Table 18: Auxiliary Results
Programming Instrument Descriptive Statistics Mean Number of
Commands Used Per Successful Program on the Reuse of
Subprocedures Programming Instrument

Test Problems:	Prob. 1	Prob. 2	Prob. 3	Prob. 4	Prob.5
Experimental ^a	17.7	19.8	27.8	12.3	39.0
Control ^b	18.0	20.0	28.1	14.5	43.4

^aLogo systematically guided toward analogical reasoning.

^bLogo taught in a traditional, exploratory approach.

Table 19: Auxiliary Results
Programming Instrument Descriptive Statistics Percent of Treatment
 Group Using Variables and Recursion Within the Reuse of
 Subprocedures Programming Instrument

	Variables	Recursion
Experimental ^a	87.5%	52.8%
Control ^b	83.3%	50.0%

^aLogo systematically guided toward analogical reasoning.

^bLogo taught in a traditional, exploratory approach.

Table 20: Auxiliary Results
Correlations Between Outcome Variables Correlations of the Cognitive Ability Test - Nonverbal Battery With Selected Programming Variables on the Reuse of Subprocedures Programming Instrument

Correlations With Cognitive Ability Test - Nonverbal Battery

	Reuse of Subproc.	Number of Prob. Correct	Use of Variables	Use of Recursion
Total Sample (N=144)	.263 (p<.001)***	.414 (p<.001)***	.200 (p<.008)**	.225 (p<.003)**

*** Signifies significance at the .001 level.

** Signifies significance at the .01 level.

CHAPTER V: DISCUSSION OF RESULTS

In this chapter the results of a study designed to investigate the potential of guided Logo programming instruction for use in the development and transfer of analogical reasoning are interpreted. Discussion will be divided into six sections: 1) a brief summary of the study, 2) an examination of results relating to hypothesized far transfer effects, 3) an examination of results relating to hypothesized near transfer effects, 4) an examination of results concerning basic LogoWriter comprehension, 5) the implications suggested by auxiliary descriptive statistics, and 6) a summary of conclusions, and recommendations for further research.

Summary of the Study

The goals for the study were to investigate two potential effects of incorporating systematic analogical reasoning training within guided Logo programming instruction. The first goal was to investigate the far transfer effects of such instruction on general analogical reasoning development, as measured by a test associated with general analogical reasoning. The second goal was to investigate the near transfer effects of such instruction on a related and important computer programming skill - the ability of the student to reuse subprocedures between programming problems.

Two bodies of research were tapped in pursuing these goals:

1) research involving the training of analogical reasoning, and
2) research involving the development of cognitive skills from programming. The study was structured to contribute to both of these areas by providing a methodology for empirically investigating analogical reasoning training in guided Logo programming. Thus, contribution to the search for potential methods to instruct general analogical reasoning was targeted by focusing on guided Logo programming as one possible method; and contribution to research on the development of general cognitive skills from programming was targeted by examining analogical reasoning as one particular skill.

To provide general analogical reasoning training within a guided programming environment, this study incorporated Swan and Black's three pedagogical components for the effective transfer of cognitive skills from programming. These components involved a focus on the specific skill, direct instruction of the skill, and a mediational approach to teacher/student interaction. Each of these transfer components was emphasized in the guided programming instruction. To incorporate the first component, the instruction focused on analogical reasoning rather than on the programming activity itself. To incorporate the second component, Sternberg's component processes of analogical reasoning were used as a framework for direct instruction. Finally, to incorporate the third component, detailed activity sheets

were utilized to facilitate teacher/student interaction in each class meeting.

Using a post-test only control group design, students were randomly placed in one of the two treatment groups. The experimental group experienced guided Logo programming instruction, whereas the control group experienced more traditional exploratory Logo programming instruction. Both groups received the same instructional content, with only the instructional treatment delivering that content varied between the guided and traditional Logo programming. Measures of transfer were operationally defined to be student scores on the Cognitive Ability Test - Nonverbal Battery, representing far transfer of learning, and a student reuse of subprocedures score on a constructed programming test, representing near transfer of learning. A multiple choice basic comprehension test was also administered to indicate relative comprehension of the LogoWriter language between treatment groups.

Two directional hypotheses were generated for the study. Hypothesis one predicted a higher group mean on the Cognitive Ability Test - Nonverbal Battery for the experimental group than for the control group. Similarly, hypothesis two predicted a higher mean reuse of subprocedures for the experimental group. These hypotheses, along with group means for the comprehension test, were tested by use of standard t-tests. To determine if additional independent variables were interacting with the treatment, further auxiliary analyses

were also performed, using the analysis of variance statistical procedure. Auxiliary data also included additional descriptive statistics and correlations, to further enhance the investigative nature of the study and to verify study assumptions.

Results for the study were reported in the previous chapter. These results included data from statistical tests of each of the two study hypotheses, and additional descriptive statistics and correlations. These results, and their particular implications concerning the potential of guided Logo programming instruction for use in the development and transfer of analogical reasoning, will now be considered.

A Discussion of Far Transfer Results

It was hypothesized that the experimental group, involved in Logo programming systematically guided toward analogical reasoning, would have a higher mean score on the Cognitive Ability Test - Nonverbal Battery than the control group, that was involved in more traditional Logo instruction. This prediction relied on research suggesting the potential success of analogical reasoning training in the classroom (Holyoak, 1984; Sternberg, 1977a, b), and on research suggesting that guided programming instruction could facilitate the development of specific cognitive skills (Swan & Black, 1987; Delclos, V., Littlefield, J., & Bransford, J., 1984). Also, such a prediction was encouraged by

the inherent use of analogical reasoning when examining past programs for insight into new ones (Kurland et al., 1987; Pennington, 1982).

The initial results

Using two treatment groups, one involved in Logo programming systematically guided toward analogical reasoning, and one involved in more traditional exploratory Logo programming instruction, the first study hypothesis of far transfer was statistically tested and the results reported in chapter four. As reported in that chapter, the initial t-test between treatment groups implied that no differential far transfer of learning had occurred, since mean composite scores on the Cognitive Ability Test - Nonverbal Battery were not found to be statistically different between the groups. Auxiliary analyses were then completed to further control for possible interactions of additional independent variables.

Searching for interactions

Four independent variables were systematically entered into factorial designs to further control for their possible interactive effects with instructional treatment: age, self-reported computer nervousness, gender, and college year. The continuous scores of age and self-reported computer nervousness were entered as covariates and found to be statistically significant. However, treatment effects

were still non-significant after controlling for these sources of variation. The effect of gender was also investigated, by using it as an independent factor in an analysis of variance factorial design. As reported, no main effect, or interactive effect, was found for the independent variable of gender. Finally, student year in college was entered as an independent factor in the factorial designs, while still controlling for computer nervousness. As reported, a statistically significant interactive effect was found between instructional treatment and a student's year in college, $p < .05$.

The interaction with year in college

It is interesting to note the pattern in this study between a student's year in college and instructional treatment. Descriptive statistics indicated that freshmen had performed best in the guided Logo programming instruction, and juniors and seniors had performed best in the traditional exploratory Logo instruction, with sophomores performing relatively equally between instructional treatments. This disordinal interaction is expressed by Figure 2. When individual analysis of variance tests were run for each specific college year, only the freshmen and junior groups were statistically significant within their own group.

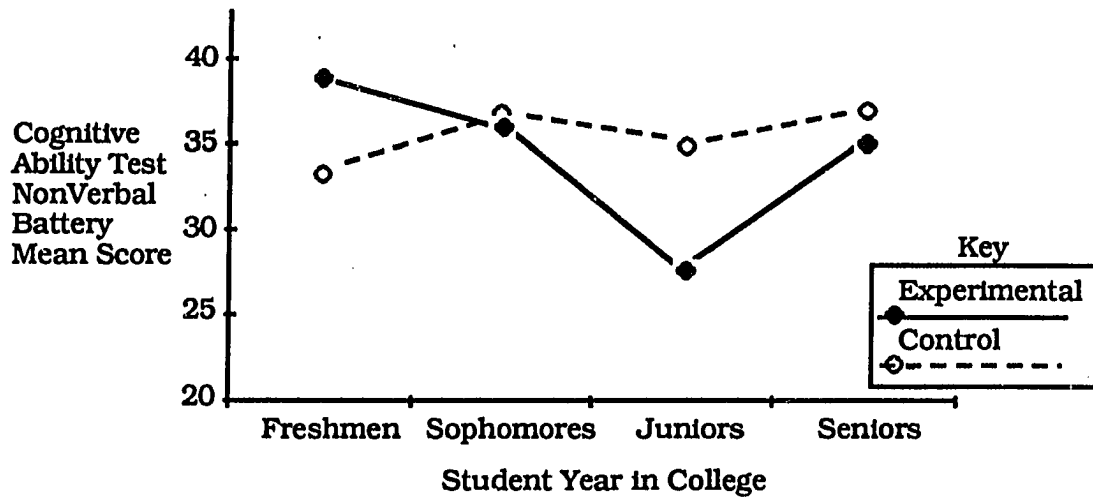


FIGURE 2. Interaction of college year with instructional treatment

There is no doubt a mix of general characteristics that might distinguish a college freshmen from a college junior. Whether college juniors actually differ from college freshmen in specific characteristics of ability, achievement, motivation, etc., is difficult, and possibly impossible, to say. However, college juniors do have the advantage of at least two years in formal college instruction, and hold the inherent benefits of experiencing those two years. Freshmen, on the other hand, are relatively academically inexperienced, and are only beginning to experience learning at the college level. It is important to note that most of the freshmen used in this study were probably within the first few weeks of their college experiences, as this study began in the first few weeks of the fall semester. Since no differences in treatment effects were found for college sophomores, quite different results may have been achieved had this study been implemented in the spring

semester, when most freshmen would have had a semester of college before beginning the instructional treatments.

Relation to ability interactions

The observed pattern of interaction between a student's year in college and instructional treatment contains similarities to that found in many studies examining the interaction of a student's general reasoning ability with instructional treatment (see Snow, 1980, Wittrocks, 1974). As suggested by Wittrock:

"Students with high general reasoning scores profit more from a treatment in which the organization and structuring essential to generative processing is left to the individual. Students with low general reasoning scores profit more from a fully elaborated treatment, one that explicitly provides the organizational structure that relates new information to previous experience" (p. 190).

It is interesting to consider whether general reasoning ability played a significant part in the differential performance for college freshmen and juniors when exposed to the different instructional treatments. If general reasoning ability did play a role, it seems likely that such general ability would be of a special type that was "evolving"

through college learning experiences, providing an inherent difference between the general reasoning ability of freshmen and juniors. This seems especially likely when considering that students were randomly assigned to the experimental treatments, statistically equating treatment groups in the more stable aspects of student general ability.

Some possible insight into this situation is found in studies investigating differences in "crystallized ability" and "fluid ability" (Snow, 1980, Hart, 1986). Crystallized ability is associated with reasoning tasks drawing on verbal knowledge, reading comprehension, and prior achievement. It tends to evolve and develop with age and experience. Fluid ability, in contrast, relates to reasoning tasks which draw little on prior achievement, but encompass the ability to deal with new and relatively different processing tasks. Fluid ability usually encompasses the more stable aspects of general ability, and is somewhat resistant to change.

Since improvement in crystallized ability is often associated with experience (Hart, 1986, Baltes & Schaie, 1982, Snow 1980), it would seem that college juniors may have increased their crystallized ability through their initial two years of college, differentiating them from the introductory freshmen. Fluid ability differences between the freshman and juniors may have played lesser a part in the observed treatment interaction, due to the higher stability of this construct and the incorporation of randomization within the study design.

The natural evolved difference between college juniors and freshmen in crystallized reasoning ability, therefore, may have been partially responsible for the observed difference in reactions to the experimental treatment. Thus, college juniors may have found the analogical reasoning instruction to be in conflict with the use of their personal reasoning strategies, developed through general college experience; creating a reduced performance on the Cognitive Ability Test. In contrast, when juniors were allowed to practice their own reasoning strategies developed through experience, as within the control treatment, their performance on the Cognitive Ability Test was facilitated. The reverse may have been true for college freshmen. Since their crystallized reasoning ability was less established, the formal analogical reasoning strategy offered by the experimental treatment may have helped to facilitate performance on the general analogical reasoning instrument. Furthermore, since personal reasoning strategies of the freshmen were less developed, the freedom to use these less efficient strategies, as offered by the control group, may have operated to hinder group performance on the instrument.

It is important to note that seniors also did slightly better in the control treatment, although this difference was not statistically significant, as it was with the junior subgroup. It would appear that seniors were not as heavily influenced by the instructional treatment as were the juniors, although the direction of influence was consistent between the groups. It is difficult to say why seniors did not react at

least as significantly as juniors to the instructional treatment. It may be that seniors, close to the end of their college experiences, put less emphasis on the class in general, and thus minimized the effects of differing instructional treatments. Or possibly, seniors who enroll in a such a low level course, so late in their college experiences, may be substantially different from the typical senior enrolled in higher level courses. Often, it is not uncommon for such seniors to take a freshmen level class as pass/fail, and that may have been the situation for some of the seniors in the present study. In any case, it is apparent from the observed interaction in this study that a year difference in college level may be associated with substantial differences in treatment effects.

A tentative conclusion of far transfer

Any conclusions should be considered tentative in investigative research of this type, but it would seem apparent that some far transfer of learning, within an interactional context, did take place in this study. The guided Logo programming instruction did facilitate analogical reasoning performance on the Cognitive Ability Test - Nonverbal Battery for freshmen in the experimental sample. However, such instruction hindered performance for juniors, slightly but did not significantly hinder performance for college seniors, and had no effect on college sophomores. Thus, the first hypothesis for the study was

supported for the college freshmen subgroup, but not supported for college sophomores, juniors, or seniors.

The observed interaction in this study would seem similar to studies reporting interactions for general ability. Although it is difficult to say what specific cognitive aspects differentiate a college junior from a college freshman, it would seem that differences in crystallized reasoning ability offer at least one avenue for an explanation of the interaction experienced in this study.

A Discussion of Near Transfer Results

This study also hypothesized that the experimental group, involved in Logo programming systematically guided toward analogical reasoning, would have a higher mean reuse of subprocedures on a constructed programming test than the control group, involved in more traditional Logo instruction. This prediction relied on research suggesting analogical reasoning is inherent in the programming process (Kurland et al., 1987; Mann, 1986; Pennington, 1982), and that a student's reuse of subprocedures is related to their success on an analogical reasoning task (Clement et al., 1986). This hypothesis was also supported by evidence that guided programming instruction is often more effective in developing specific programming and cognitive skills (Leron, 1985, Swan & Black, 1987), than less directed programming instruction.

The initial results

Using two treatment groups, one involved in Logo programming systematically guided toward analogical reasoning, and one involved in more traditional exploratory Logo programming instruction, the second study hypothesis of near transfer was tested statistically and the results reported in chapter four. As discussed in that chapter, initial variance in the reuse of subprocedures scores differed statistically between treatment groups, with the larger variance of scores existing within the guided programming treatment. This difference necessitated that raw scores be transformed logarithmically to meet the equal variance assumptions of further statistical tests.

As reported in Chapter Four, the initial t-test of transformed scores indicated that the group means for reuse of subprocedures were not statistically different. This result implied that no differential effects in near transfer of learning had occurred between treatment groups. Auxiliary analyses were then completed to further control for possible interactions.

Searching for interactions

Similar to the auxiliary procedures for hypothesis one, four independent variables were systematically entered into factorial designs to statistically control for their possible interactive effects with instructional treatment: age, self-reported computer nervousness,

gender, and college year. The continuous scores of age and a self-reported computer nervousness were entered as covariates and found to be non-significant sources of variation. The effect of gender was also investigated, by using it as an independent factor in an analysis of variance factorial design. Although no interaction with gender was found, a main effect for gender approached significance, with females reusing slightly more subprocedures than males. Also, within the males subgroup itself, males in the experimental group had reused slightly more subprocedures than males in the control group; however, this difference also did not achieve statistical significance. Finally, student year in college was also entered as an independent factor in the factorial designs, with no evidence of interaction or main effects. Thus, auxiliary analyses for the second hypothesis, controlling for the effects of four additional independent variables, also did not find any statistical evidence of near transfer effects.

A possible explanation

The results of the study indicating that there had been essentially no near transfer of learning for the guided programming group was at first surprising. The use of the Sternberg component processes to directly reference prior problems in the construction of new programs, would seem to generally encourage the direct reuse of subprocedures from the past problem. By systematically focusing on a specific earlier problem, direct use of subprocedures in that problem

would seem to be made more apparent and cognitively available to the student programmer. The non-significant results for the near transfer hypothesis also seemed inconsistent with significant interaction results for the first hypothesis of the study, since no further interaction was found for a student's year in college. However, upon review of Salomon and Perkins discussion of the "high road" and "low road" transfer mechanisms (1987), a tentative explanation becomes apparent.

In discussion by Salomon and Perkins (pp. 151-153), "high road" orientation for transfer seeks to achieve transfer of learning by use of mindful abstraction of a skill, so as to view it in a more general sense and as useful to other domains. This "high road" orientation is often associated with the far transfer of learning into domains different from those in which the skill is initially practiced. In contrast, "low road" transfer orientation seeks to achieve transfer by extensive repetition and automation of a skill, and is often associated with very near or same domain transfer. Salomon and Perkins make the point that extensive practice in programming would be unnecessary for far transfer to other domains, as long as a vigorous high road transfer orientation was present. In near domain transfer, however, more extensive practice of a skill, using a low road transfer orientation, may be necessary, encouraging the skill to become fairly automaticized within the content domain.

It seems that such transfer mechanisms were possibly in operation in this study. The short duration of the study may have made near domain transfer difficult; as repetition of the specific process of reusing subprocedures was not emphasized by the study, and instructional time may not have been extensive enough for this skill to become naturally automated. However, since far transfer of learning was directly emphasized by use of a high road transfer orientation, adequate time and practice may have been available for it to be achieved. This rationale would help to explain why significant results, associated with college year interaction, were found for the first hypothesis, representing far transfer, but not for the second hypothesis, representing near transfer.

It is important to note that the experimental treatment systematically emphasized far transfer of learning by incorporation of the pedagogical transfer components discussed by Swan and Black (1987). Thus, the active attention of students within the experimental group was continually focused on the general problem solving nature of analogical reasoning, and not on the programming process itself. This emphasis, although encouraging far transfer to geometric analogy problems present on the Cognitive Ability Test - Nonverbal Battery, possibly did little to encourage near transfer to reuse of subprocedures between programming problems. It would seem that near transfer of learning was not facilitated in this study by a high road instructional treatment that systematically focused on far transfer of learning.

It is interesting to note, however, that the experimental treatment group did have a significantly higher variance than the control group. This finding would imply that the experimental treatment may have had some differential effects for some students. A post-hoc search for interaction did not identify any pattern for this difference in the experimental scores. It may well be that additional independent variables were operating interactively to spread out the reuse of subprocedure scores for some students.

A tentative conclusion for near transfer

Conclusions for hypothesis two, as with conclusions associated with hypothesis one, must be considered tentative due to the general and investigative nature of the study. The guided Logo programming instruction, although providing an increased variance in scores, did not statistically improve analogical reasoning performance related to the increased reuse of subprocedures between programming problems. Thus, the second hypothesis for the study was not supported.

This result seems consistent with discussions of "high road" and "low road" paths to transfer of learning as expressed by Salomon and Perkins (1987). The repetition needed for adequate near transfer of learning may not have been available due to the relatively short duration of the study. The "high road" transfer emphasis of this study, although achieving some far transfer, may have done very little to

encourage and differentiate near transfer of learning, as represented by reuse of subprocedures within the programming domain.

A Discussion of Basic LogoWriter Comprehension

A LogoWriter basic comprehension test was developed and administered in the study to determine relative comprehension of basic commands and concepts in the LogoWriter language operating as instructional content. This test was necessary to confidently interpret any differences found in the measurement instruments representing transfer of learning. Such a test was especially needed to provide confidence in any results found for the reuse of subprocedures instrument; a test that attempted to measure a higher level programming concept that might be easily influenced by lower level differences in basic comprehension of the LogoWriter language. Thus, a multiple choice comprehension test was developed, locally standardized, and administered in the study.

Discussion of results for the test

As reported in Chapter Four, the mean scores for the basic LogoWriter comprehension test were not statistically different between treatment groups. Since the content of the test was tied directly to instructional objectives taught in both treatment groups, this result implied that both treatment groups received a relatively equal

understanding of basic commands and concepts in the LogoWriter language, operating as the instructional content.

The statistically equal achievement of both treatment groups on the LogoWriter basic comprehension test was seen as an encouraging result. This study investigated the relative far and near transfer effects of two different instructional techniques teaching the same instructional content. If basic comprehension of that content had differed significantly, then conclusions about transfer, especially near transfer in the programming domain, would need to consider that lower level comprehension differences might be responsible for higher level transfer results. Such a situation would have also suggested the possibility that attempts to teach the same instructional content to both treatment groups had not been successful. Fortunately, however, treatment means were not found to differ significantly for this test, and it seems reasonable to conclude that the study had been relatively successful in focusing on hypothesized transfer differences, rather than on lower level comprehension differences, between the instructional treatments.

It is also interesting to note that the relative comprehension of the instructional content between groups was statistically equivalent even though the guided Logo group generally spent less time on the computer than the control group, using more traditional exploratory Logo instruction. Although this was not a focus of the study, such a result suggests the possibility that programming instruction guided

toward analogical reasoning may be able to use less on-line time than more traditional exploratory Logo programming, and still achieve at least an equivalent understanding of basic programming concepts. However, research that more directly compares on-line times between groups, and includes tests of the retention of instructional content, as well as its immediate comprehension, would be needed before such a conjecture could be confidently made.

Implications of Auxiliary Descriptive Statistics

To support the investigative nature of the study, additional descriptive statistics were gathered and reported in the study. The descriptive statistics summarized group programming performance on the constructed programming test related to other aspects of programming other than the reuse of subprocedures between programming problems. Implications of these statistics will now be discussed.

Discussion of programming descriptive statistics

The constructed programming test, used in the investigation of hypothesized group differences for the reuse of subprocedures between programming problems, was also scored to reflect group performance in four other programming aspects. These were: 1) the number of problems successfully programmed, 2) the number of

commands used per successful problem, 3) the use of variables, and 4) the use of recursion. These statistics were reported to help provide insight into results related to the reuse of subprocedures between programming problems, and to help recommend further research.

As reported in Chapter Four, although the experimental group had done slightly better in all four sets of additional descriptive measures, most of these scores were relatively close between treatment groups. Two notable differences did become apparent, though. First, in the three problems using rectangles, the experimental group had an average of 10% better success than the control group. Secondly, in the last two test problems, experimental students who had successfully programmed these problems, used an average of two to four commands less in their programs than did students in the control group.

Possible implications It is interesting to note that both differences occurred in the three problems of greatest probable difficulty for the students, the problems using rectangles. This observation suggests that the difficulty level of the test problems may have hindered effective discrimination between the programming capabilities of the two groups. Although the programming test itself seemed to be viewed as quite difficult by the students, the test problems themselves may not have been difficult enough to demand that the students draw on careful problem solving strategies. When problems did begin to get more difficult, as with the three rectangle

problems, the difference between the experimental and control groups seemed to become more apparent.

The failure to find a difference between treatment groups in the reuse of subprocedures, representing near transfer effects, may then also be a function of inadequate test problem difficulty, as well as inadequate instructional time or repetition, as discussed earlier. The individual test problems may have lacked sufficient difficulty to encourage the guided Logo group to attempt to apply their instructed analogical reasoning strategy. Some students may have purposefully chosen an approach of creating completely new programs for each particular problem, because it seemed easier than trying to reuse previous subprocedures. Test problems may not have necessitated that students carefully encode the various characteristics of the problem, which is so important in the general analogical reasoning process. The loss of such a reasoning step would have greatly hindered a student's tendency to reuse subprocedures between the test problems. It seems apparent, then, from the additional descriptive statistics on the constructed programming test, that programming problems of greater difficulty than those used in this study may be necessary to effectively elicit the instructed analogical reasoning strategy.

A more promising evaluative approach may be to use fewer, but more difficult and carefully structured problems. Perhaps a pair of problems, or sets of pairs, carefully designed to share structural aspects, could focus more directly on the analogical reasoning

processes involved in a student drawing insight from one problem to the other. Such pairings could emphasize specific aspects of the problems that related to particular components of the analogical reasoning process. Thus, one pair might emphasize aspects related to encoding, another pair might emphasize aspects related to inferring, etc. Such a systematic focus on each of the individual component processes of analogical reasoning would greatly contribute to knowledge about how these components operate within the programming domain, and how they might be facilitated and improved in student programmers.

Summary of Conclusions and Research Recommendations

This study investigated the potential of guided Logo programming instruction for use in the development and transfer of analogical reasoning. Investigation of this potential focused on two possible treatment effects: 1) the far transfer of instruction, as measured by a test associated with general analogical reasoning, and 2) the near transfer of instruction, as measured by a constructed programming test targeting the reuse of program subprocedures.

Conclusions:

Although of an exploratory nature, necessitating that conclusions be considered tentative, results from this study indicate the following:

1. Guided Logo programming instruction significantly facilitated general analogical reasoning performance for college freshmen, as measured by the Cognitive Ability Test - Nonverbal Battery, and as compared to traditionally instructed Logo programming.
2. Guided Logo programming instruction significantly hindered general analogical reasoning performance for college juniors, as measured by the Cognitive Ability Test - Nonverbal Battery, and as compared to traditionally instructed Logo programming.
3. Significant Interaction was found for the guided Logo programming instruction and a student's year in college.
4. Guided Logo programming instruction did not significantly increase students' reuse of subprocedures between programming problems, as measured by a constructed programming test, and as compared to traditionally instructed Logo programming. However, a higher statistical variance in reuse of program subprocedures was observed for the guided instructional group.
5. The constructed reuse of subprocedures programming test, as modified from previous research, may not have contained problems of

sufficient difficulty to effectively and fully elicit the instructed analogical reasoning strategy in student completion of the test.

Recommendations for further research

Based on this study, the following recommendations for further research are suggested:

1. Study results seem to indicate that guided Logo programming, as structured in this study, may be differentially effective across various student characteristics for the development and far transfer of general analogical reasoning. Further research focusing on specific interactions with student characteristics, such as crystallized and fluid ability, would appear warranted.
2. Study results seem to indicate that reuse of subprocedures may not be an adequate "stand alone" representation of the analogical reasoning near transfer effects of guided Logo programming. A more comprehensive approach, focusing on a variety of programming aspects, may be more conducive to the investigation of near transfer effects.
3. Study results seem to suggest that the reuse of subprocedures programming test, as structured in this study, may not be appropriate for looking at near transfer effects of the guided programming

instruction. A more promising evaluative approach may be to use pairs of more difficult, and carefully structured programming problems, that can effectively elicit analogical reasoning, and can be linked with specific component processes of the skill.

4. Study results suggest the possibility that programming instruction guided toward analogical reasoning may be able to utilize less on-line time than more exploratory programming instruction in the learning of basic programming concepts. Further research that more directly compares group on-line times, comprehension of instructional content, and actual retention of that content, would seem appropriate.

5. Further investigations of guided Logo programming, as structured in this study, should include a variety of age groups and grade levels. It appears that guided programming instruction, targeted at analogical reasoning, may have substantially different effects for students of varying ages and levels of formal education. Students younger than those used in the present study would seem to be especially appropriate for further research.

Concluding Remarks

In this study, the potential of guided Logo programming for use in the development and transfer of analogical reasoning was investigated.

It was an exploratory study seeking to contribute to the ongoing search for possible classroom methods to instruct general cognitive skills, by focusing on analogical reasoning as one specific skill, and by using guided Logo programming as one particular method.

Although further research needs to be completed, guided Logo programming does seem to offer a powerful instructional tool for teaching general analogical reasoning strategies to some students. Computer programming languages themselves, especially Logo, appear to offer a flexible and explicit problem solving medium by which analogical reasoning strategies can be effectively discussed and illustrated.

It would appear that the "potential" of guided Logo programming instruction for use in the development and transfer of analogical reasoning is an exciting one, and worthy of continued research. Such research may be all the more important as we enter an expanding age of information, and attempt to meet the educational challenges of that age. The active search for effective ways to instruct general analogical reasoning would seem especially imperative for the well-being of today's students who are citizens of the information age. Analogical reasoning is a problem solving skill that can assist those students in looking confidently ahead, by carefully looking back at what they already know; an important skill in a time where there is indeed so much to know.

BIBLIOGRAPHY

- Adelson, B. (1981). Problem solving and the development of abstract categories in programming languages. Memory & Cognition, 9(4), 422-433.
- Alderton, D. (1985). Individual differences in process outcomes for verbal analogy and classification solution. Intelligence, 9(1), 69-85.
- Alexander, A. White, C. Haensly, P., & Crimmins-Jeanes, M. (1987). Analogy training: A study of the effects of verbal reasoning. Journal of Educational Research, 24(3), 77-80.
- Anderson, J. R. (1982). Acquisition of cognitive skill. Psychological Review, 89, 369-406.
- Anderson, J. R., Greeno, J. R., Kline, P. J., & Neves, D. M. (1981). Acquisition of problem solving skill. In J. R. Anderson (Ed.), Cognitive Skills and Their Acquisition. Hillsdale, NJ: Erlbaum.
- Anderson, R. C. (1984). Role of the reader's schema in comprehension, learning, and memory. In R. C. Anderson, J. Osborn, & R.J. Tierney (Eds.), Learning to read in American schools. Hillsdale, NJ: Erlbaum Publishers.
- Arons, A. B. (1984). Computer based instructional dialogs in science courses. Science, 224, 1051-1056.
- Baltes, P. B., & Schale, K. W. (1982). Aging and IQ--the myth of the twilight years. In S. H. Zarit (Ed.), Readings in aging and death: Contemporary perspectives. New York: Harper Row.
- Becker, H. J. (1987). The importance of a methodology that maximizes falsifiability: Its applicability to research about Logo. Educational Researcher, 16(5), 11-16.
- Bisanz, J. (1984). Interpretation of instructions: A source of individual differences in analogical reasoning. Intelligence, 8(2), 161-177.
- Borg, W. R. (1983). Educational Research: An Introduction. New York, New York: Longman Publishing.

- Brooks, R. E. (1977). Toward a theory of the cognitive processes in computer programming. International Journal of Man-Machine Studies, 9, 737-751.
- Brooks, L. & Dansereau, D. (1986). Effects of structural schema and text organization on expository prose processing. Journal of Educational Psychology, 75, 511-520.
- Brown, J. S., & Burton, R. (1978). Diagnostic models for procedural bugs in basic mathematical skills. Cognitive Science, 2, 155-192.
- Cambre, M., & Cook, D. (1985). Computer anxiety: definition, measurement, and correlates. Journal of Education Computing Research, 1(1), 37-54.
- Christ-Whitzel, J., & Hawley-Winne, B. (1976). Individual differences and mathematics achievement: An investigation of aptitude-treatment interaction in evaluation of three instructional approaches. ERIC ED 129 868.
- Clement, C., Kurland, D. Mawby, R., & Pea, R. (1986). Analogical reasoning and computer programming. Journal of Educational Computing Research, 2(4), 473-485.
- Clements, D. H. (1987). Longitudinal study of the effects of Logo programming on cognitive abilities and achievement. Journal of Educational Computing Research, 3(1), 73-94.
- Clements, D. H. (1986). Effects of Logo and CAI environments on cognition and creativity. Journal of Educational Psychology, 78(4), 309-318.
- Clements, D. H. (1985). Logo programming: Can it change how children think? Electronic Learning, 28, 74-75.
- Clements, D. H., Gullo, D. (1984). Effects of computer programming on young children's cognition. Journal of Educational Psychology, 76, 1050-1059.
- Clements, D. H., & Merriman, S. (1987). Componential developments in Logo programming environments. Paper presented at the American Educational Research Association, Washington D. C.

- Cormier, S.; & Hagman, J. (1987). Transfer of Learning. Contemporary Research and Applications. San Diego, California: Academic Press.
- Cory, S., & Walker, M. (1985). Logo works: Lessons in Logo. Cambridge, Massachusetts: Terrapin Logo.
- Cronbach, L., & Snow, R. (1977). Aptitudes and Instructional Methods. New York, New York: Irvington Publishers.
- Davidson, R. (1976). The role of metaphor and analogy in learning. In J. Levin, and V. Allen (Eds.), Cognitive Learning in Children: Theories and Strategies. New York, New York: Academic Press.
- De Corte, E., & Verschaffel, L. (in press). Effects of computer experience on children's thinking skills. Journal of Structural Learning.
- De Leeuw, L. (1983). Teaching problem solving: An ATI study of the effects of teaching algorithmic and heuristic solution methods. Instructional Science. 12(1), 1-48.
- Degelman, D., Free, J., Scarlato, M., Blackburn, J., & Golden, T. (1986). Concept learning in preschool children: Effects of a short-term Logo experience. Journal of Educational Computing Research. 2, 199-205.
- Dewey, J. (1933). How We Think: A Restatement of the Relation of Reflective Thinking to the Educative Process, Lexington, Massachusetts: D.C. Heath.
- Delclos, V., Littlefield, J., & Bransford, J. (1984). Teaching thinking through Logo: The importance of method. Technical Report 1.2, Vanderbilt University Learning Technology Center, Nashville, TN. Eric ED 262 756.
- Doyle, W. (1983). Academic work. Review of Educational Research. 53, 159-199.
- Ekstrom, R. B., French, J. W., & Harmon, H. (1976). Manual for Kit of Factor-Referenced Cognitive Tests. Princeton, NJ: Educational Testing Service.

- Evans, T. G. (1968). A program for the solution of geometric analogy intelligence test questions. In M. Minsky (Ed.), Semantic Information Processing, Cambridge, Massachusetts: MIT Press.
- Federico, P. (1978). Accommodating instruction to student characteristics: trends and issues. Navy Personnel Research and Development Center Report, San Diego, CA. ERIC ED 165 792.
- Feurzig, W., Horowitz, P., & Nickerson, R. (1981). Microcomputers in Education. Cambridge Massachusetts: Bolt, Beranek, and Newman.
- Fredericksen, N. (1984). Implications of cognitive theory for instruction in problem solving. Review of Educational Research, 54(3), 363-407.
- Gagne', R. M. (1986). Instructional Technology: The Research Field. Journal of Instructional Development, 8(3), 7-14.
- Garner, R., Wagoner, S., & Smith, T. (1983). Externalizing question-answering strategies of good and poor comprehenders. Reading Research Quarterly, 18, 439-447.
- Gentner, D. (1982). Structure mapping: A theoretical framework for analogy. Cognitive Psychology, 7, 155-70.
- Gentner, D., Stevens, A. L. (1983). Mental Models. Hillsdale, NJ: Erlbaum Publishers.
- Gick, M., & Holyoak, K. (1980). Analogical problem solving. Cognitive Psychology, 12(3), 306-355.
- Gick, M., & Holyoak, K. (1983). Schema induction and analogical transfer. Cognitive Psychology, 15, 1-38.
- Goldman, S. R., & Bisanz, J. (1980). Understanding the development of analogical reasoning ability. Paper presented at AERA, Boston, MA. ERIC ED 186 132.
- Goldman, S. R., Pellegrino, J. W., Parseghian, P., & Sallis, R. (1982). Developmental and individual differences in verbal analogical reasoning. Child Development, 53(2), 550-559.
- Greeno, J. G. (1978). Natures of problem solving abilities. In K. W. Estes (Ed.), Handbook of Learning and Cognitive Processes. Hillsdale, NJ: Erlbaum.

- Gugerty, L., & Olson, G. (1986) Comprehension differences in debugging by skilled and novice programmers. In E. Soloway and S. Iyengar (Eds.), Empirical Studies of Programmers. Norwood, New Jersey: Ablex Publishing.
- Guilford, J. P. (1967). The Nature of Human Intelligence. New York, New York: McGraw Hill.
- Halpern, D. (1987). Analogies as a critical thinking skill. In D.E. Berger, K. Pezdek, W.P. Banks (Eds.) Applications of Cognitive Psychology: Problem Solving, Education, and Computing. Hillsdale, NJ: Erlbaum Publishers.
- Hart, R. (1986). The effect of fluid ability, visual ability, and visual placement within the screen on a simple concept task. Paper presented at AECT convention, Las Vegas, NV. ERIC ED 267 774.
- Hayes-Roth, B., & Hayes-Roth, F. (1979). A cognitive model of planning. Cognitive Science, 3, 275-310.
- Hesse, M. B. (1966). Models and Analogies in Science. Notre Dame: University of Notre Dame Press.
- Holton, B. (1982). Attribute treatment interaction research in mathematics education. School Science and Mathematics, 82(7), 593-602.
- Holyoak, K. (1984). Analogical thinking and human intelligence. In R. J. Sternberg (Ed.), Advances in the Psychology of Human Intelligence. Hillsdale, NJ: Erlbaum Publishers.
- Horton, J., & Ryba, K. (1986). Assessing learning with Logo: A pilot study. The Computing Teacher, 14(1), 24-28.
- Howe, J., O'Shea, T., & Plane, F. (1979). Teaching mathematics through Logo programming: An evaluation study. In R. Lewis and E. D. Tagg (Eds.), Computer Assisted Learning: Scope, Progress, and Limits. Amsterdam, Holland: North Holland Press.
- Hunt, M. (1982). The Universe Within. New York, New York: Simon and Schuster.

- Khayrallah, M., & Van Den Meiraker, M. (1987). Logo programming and the acquisition of cognitive skills. Journal of Computer-Based Instruction, 14(4), 133-137.
- Kurland, D., Clement, C., Mawby, R., & Pea, R. (1987). Mapping the cognitive demands of learning to program. In R. Pea and K. Sheingold (Eds.), Mirrors of Minds. Norwood, New Jersey: Ablex Publishing, 103-127.
- Kurland, D., & Pea, R. (1983). Children's mental models of recursive Logo programs. Journal of Educational Computing Research, 1, 235-243.
- Lawson, A. (1982). Formal reasoning, achievement, and intelligence, An issue of importance. Science Education, 66, 77-83.
- Leron, U. (1985). Logo today: Vision and reality. Computing Teacher, 12(5), 26-32.
- Mann, R. J. (1986). The effects of Logo computer programming on problem solving abilities of eighth grade students. Dissertation Abstracts International, 8619040.
- Markman, E. M. (1977). Realizing that you don't understand: A preliminary investigation. Child Development, 48, 986-992.
- Mathison, C., & Allen, B. (1987). The effect of stories and diagrams on the solution on an analogous problem. Paper Presented at the Annual Convention of the Association for Educational Communications and Technology, Atlanta, GA. ERIC ED 285 549.
- Mawby, R. (1984). Structured interviews on children's conceptions of computers. Technical Report 19. Bank Street College of Education, New York, New York.
- Mayer, R. (1987). Learnable aspects of problem solving: Some examples. In D. Berger, K. Pezdek, and W. Banks (Eds.), Applications of Cognitive Psychology: Problem Solving, Education, and Computing. Hillsdale, NJ: Lawrence Erlbaum.
- Mayer, R. (1979). Can advance organizers influence meaningful learning? Review of Educational Research, 49, 371-383.

- Mayer, R., Bayman, P., & Dyck, J. (1987). Learning programming languages: research and applications. In D. Berger, K. Pezdek, and W. Banks (Eds.) Applications of Cognitive Psychology: Problem Solving, Education, and Computing. Hillsdale, NJ: Lawrence Erlbaum Publishers.
- McConaghy, J., & Kirby, N. (1987). Analogical reasoning and ability level: An examination of R.J. Sternberg's componential method. Intelligence, 11(2), 137-159.
- McGee, J. (1987). Curriculum for the information age: An interim proposal. In M. A. White (Ed.), What Curriculum for the Information Age? Hillsdale, NJ: Lawrence Erlbaum Associates.
- McKinnon, J., & Renner, J. (1971). Are colleges concerned with intellectual development? American Journal of Physics, 39(9), 1047-1052.
- McLeod, D., & Adams, V. (1980). Aptitude treatment interaction in mathematics instruction using expository and discovery methods. Journal for Research in Mathematics Education, 11, 225-234.
- McLeod, D., & Adams, V. (1979). The role of cognitive style in the learning of mathematics. Technical Report San Diego State University, ERIC ED 196 684.
- McLeod, D., & Briggs, J. (1980). Interactions of field independence and general reasoning with inductive instruction in mathematics. Journal for Research in Mathematics Education, 11, 94-103.
- Moore, J., & Newell, A. (1973). How can MERLIN understand? In L. W. Gregg (Ed.), Knowledge and Cognition, Hillsdale, NJ: Lawrence Erlbaum Associates.
- Mulholland, T. Pellegrino, J., & Glaser, R. (1980). Components of geometric analogy solution. Cognitive Psychology, 12, 252-284.
- Newby, T. J., & Stepich, D. A. (1987). Learning abstract concepts: the use of analogies as a mediational strategy. Journal of Instructional Development, 10(2), 20-26.
- Newell, A., & Simon, H. (1972). Human Problem Solving. Englewood Cliffs, NJ: Prentice Hall.

- Nichols, J. A. (1981). Problem solving strategies and organization of information in computer programming. Dissertation Abstracts International, 41 (1981) 4721b.
- Nummedal, S. G. (1987). Developing reasoning skills in college students. In D.E. Berger, K. Pezdek, W.P. Banks (Eds.), Applications of Cognitive Psychology: Problem Solving, Education, and Computing. Hillsdale, NJ: Erlbaum Publishers.
- Odom, M. (1982). The effects of learning computer programming language on fifth and sixth grade student's skills of analysis, synthesis, and evaluation. Dissertation Abstracts International, 44 (1982) 64A.
- Onorato, L., & Schvaneveldt, R. (1986). Programmer/nonprogrammer differences in specifying procedures to people and computers. In E. Soloway, S. Iyengar (Eds.), Empirical Studies of Programmers. Norwood, NJ: Ablex Publishers.
- Ott, L. (1984). An Introduction to Statistical Methods and Data Analysis. Boston, MA: Duxbury Press.
- Overall, T. (1981). Learning with logo at the Lamplighter School. Microcomputing, 1, 42-47.
- Papert, S. (1972). Teaching children to be mathematicians versus teaching about mathematics. International Journal for Mathematics Education, Science, and Technology, 3, 249-262.
- Papert, S., Watt, D., diSessa, A., & Weir, S. (1979). Final Report of the Brookline Logo Project. (Logo Memo 53). MIT Artificial Intelligence Laboratory, Cambridge, MA.
- Papert, S. (1980). Mindstorms: Children, Computers, and Powerful Ideas. New York, New York: Basic Books.
- Pea, R. (1987). The aims of software criticism: Reply to Professor Papert. Educational Researcher, 16(5), 4-8.
- Pea, R. (1985). Transfer of thinking skills: Issues for software use and design. Paper presented at the National Conference of Computers and Complex Thinking, National Academy of Sciences, Washington, DC.

- Pea, R. (1983). Logo programming and problem solving. Technical Report No. 12, Bank Street College of Education, New York, New York.
- Pea, R., & Kurland, D. (1987). On the cognitive effects of learning computer programming. In R. Pea and K. Sheingold, (Eds.), Mirrors of Minds. Norwood NJ: Ablex Publishing.
- Pea, R., & Kurland, D. (1984a). On the cognitive effects of learning computer programming: A critical look. Technical Report No. 9, Bank Street College of Education, New York, New York.
- Pea, R., & Kurland, D. (1984b). Logo programming and the development of planning skills. Technical Report No. 16, Bank Street College of Education, New York, New York.
- Pea, R., Kurland, D., & Hawkins, J. (1987). Logo and the development of thinking skills. In R. Pea and K. Sheingold, (Eds.), Mirrors of Minds. Norwood NJ: Ablex Publishing.
- Pellegrino, J., & Glaser, R. (1982). Analyzing aptitudes for learning: Inductive Reasoning. In R. Glaser (Ed.), Advances in Instructional Psychology. Hillsdale NJ: Erlbaum.
- Pellegrino, J., & Glaser, R. (1979). Cognitive correlates and components in the analysis of individual differences, Intelligence, 3, 187-214.
- Pennington, N. (1982). Cognitive components of expertise in computer programming: A review of the literature. Technical Report No. 46. University of Michigan Center for Cognitive Science, Ann Arbor, Michigan.
- Perkins, D. (1985). The fingertip effect: How information-processing technology shapes thinking. Educational Researcher, 14(7), 11-17.
- Perkins, D. N., Simmons, R., & Tishman, S. (1989). Teaching cognitive and metacognitive strategies. Paper Presented at the American Educational Research Association Conference of 1989, San Francisco, California.
- Polya, G. (1957). How to solve it. Second Edition. New York, New York: Double day.

- Rumelhart, D. E., & Norman, D. A. (1981). Analogical processes in learning. In J. R. Anderson, (Ed.), Cognitive Skills and Their Acquisition. Hillsdale, NJ: Erlbaum Associates.
- Salomon, G., & Perkins, D. (1987). Transfer of cognitive skills from programming: When and how. Journal of Educational Computing Research, 3(2), 149-169.
- Schildkamp-Kundiger, E. (1982). An international review of gender and mathematics. ERIC ED 222 326.
- Snedecor G. W. & Cochran, W. G. (1980). Statistical Methods. Ames, Iowa: Iowa State University Press.
- Sneiderman, B. (1976). Exploratory experiments in programmer behavior. International Journal of Computer and Information Sciences, 5(2), 123-143.
- Sneiderman, B., & Mayer, R. (1979). Syntactic/semantic interactions in programmer behavior: A model and experimental results. International Journal of Computer and Information Sciences, 8(3), 219-238.
- Snow, R. E. (1980). Aptitude processes, aptitude, learning and instruction. In R. Snow & F. Montague, (Eds.), Cognitive Process Analysis. Hillsdale, NJ: Lawrence Erlbaum Publishers.
- Snow, R. E. (1977). Individual differences and instructional theory. Educational Researcher, 6, 11-15.
- Snow, R. E., & Lohman, D. F. (1984). Toward a theory of cognitive aptitude for learning from instruction. Journal of Educational Psychology, 76(3), 347-376.
- Stanley, J. C. & Hopkins, K. D. (1972). Educational and Psychological Measurement and Evaluation, Englewood Cliffs, NJ: Prentice Hall.
- Sternberg, R. J. (1982). Reasoning, problem solving, and intelligence. In R. Sternberg (Ed.), Handbook of Human Intelligence, Cambridge, Massassusetts: Cambridge University Press.
- Sternberg, R. J. (1980). An aptitude X strategy interaction in linear syllogistic reasoning. Journal of Educational Psychology, 72(2), 226-239.

- Sternberg, R. J. (1977a). Component processes in analogical reasoning. Psychological Review, 84, 353-378.
- Sternberg, R. J. (1977b). Intelligence, Information Processing and Analogical Reasoning: The Componential Analysis of Human Abilities. New York, New York: John Wiley and Sons.
- Sternberg, R. J., & Downing, C. (1982). The development of higher order reasoning in adolescence. Child Development, 53(1), 209-221.
- Sternberg, R. J., & Rifkin, B. (1979). The development of analogical reasoning processes. Journal of Experimental Child Psychology, 27, 195-232.
- Sternberg, R. J., Ketron, J., & Powell, J. (1982). Componential approaches to the training of intelligent performance. In D. K. Detterman & R. J. Sternberg (Eds.), How and How Much Can Intelligence Be Increased? Norwood, NJ: Abex Publishing, 155-172.
- Stonier, T. (1983). The Wealth of Information. London, England: Methuen Publishing.
- Sugrue, B. (1989). Differential Effects of High and Low Cognitive Load Instructional Methods on Recall and Transfer of Procedural Knowledge. Unpublished Doctoral Dissertation, Iowa State University, Ames, Iowa.
- Swan, K., & Black, J. (1987). The cross-contextual transfer of problem solving skills. CCT Report, 87(3), Columbia University, New York.
- Thorndike, R. L., & Hagen, E. (1986). The Cognitive Ability Test. Chicago, Illinois: Riverside Publishing.
- Thorndike, R. L., & Hagen, E. (1987a). Preliminary Examiners Manual: The Cognitive Ability Test. Chicago, Illinois: Riverside Publishing.
- Thorndike, R. L., & Hagen, E. (1987b). Technical Manual: The Cognitive Ability Test. Chicago, Illinois: Riverside Publishing.
- Tobias, S. (1976). Achievement treatment interactions. Review of Educational Research, 46(1), 61-74.

- Walker, D. (1987). Logo needs research: A response to Papert's paper. Educational Researcher, 16(5), 9-11.
- Watt, D. (1982). Logo in the Schools. Byte, 7(8), 116-134.
- White, C. S., & Alexander, P. A. (1984). Teaching analogical reasoning processes. Reading World, 24(1), 38-42.
- Wicks, R. (1986). Applying schema theory to mass media information processing: Moving toward a formal model. ERIC ED 272 870.
- Wittrock, M. C. (1974). A generative model of mathematics learning. Journal for Research in Mathematics Education, 5, 181-196.

ACKNOWLEDGEMENTS

Many people made this dissertation possible, and I would like to thank just a few of them. First, I would like to thank Dr. Ann Thompson, who advised me on this dissertation, and gave me continual advice, support, and help. Without a doubt, this dissertation would not have been a reality without her professional expertise and commitment. I would also like to thank the rest of my program of study committee, including Dr. Mike Simonson, Dr. Bill Rudolph, Dr. Rex Thomas, and Dr. Tony Netusil. A special thanks also goes to Dr. Simonson, who acted as the co-chair of my program of study committee, and for the periodic stops in his office for advice. I would also like to thank Dr. Volker, who helped make it possible for me to be at Iowa State to begin with, and Dr. Doug DeShazer, for his sound writing advice, and solutions to my periodic "Writers block".

There come to mind many others who deserve thanks, including the graduate assistants who participated as instructors in my study, including Nancy Carley, Susie Stoll, Dawn Wubben, Dave Carlsen, and Miok Lee. Each of these people have my sincere gratitude for the significant part they played in this endeavor.

I would like to especially thank those that initially helped "get me here", and gave me the confidence to attempt, and stick with such a seemingly difficult task. First, I would like to thank my father, for the courage I have drawn from him over the years, and for always being able to "get up after you're knocked down", an important skill in life as

well as graduate school. I would also like to thank my mother, for a home that always offered a chance to talk and renew perspectives. I would also like to thank my brothers and sister, for their continual encouragement and friendship, and the periodic fishing and hunting, that enabled me to forget this dissertation for a while. Finally, I would like to thank my wife Annie, for allowing me to borrow some of "our time" in working on this dissertation. It is a debt in which I will find great pleasure in repaying.

APPENDIX A: SAMPLE BACKGROUND QUESTIONNAIRE

Secondary Education 101 - Beginning of Semester Questionnaire

Please Note: This is a questionnaire given by your Secondary Education 101 instructors to help us learn a little about what interests, concerns, and backgrounds the students enrolled in this course typically have. It will be used to help us plan and improve the general instruction for future semesters, and also to help us analyze the appropriateness of the current instruction. This information will be kept strictly confidential, and will have absolutely no bearing in determining your course grade. Thanks for taking time to fill this out, and welcome to Secondary Education 101!!!

Name _____
Major: _____

Social Security No. _____
Sex: _____ Age: _____
Year in College: _____

1. Please list any high school or college computer science courses below:

High School Computer Courses:

College Computer Courses:

1) _____
2) _____
3) _____
4) _____
5) _____

1) _____
2) _____
3) _____
4) _____
5) _____

2. Please Check the programming languages you have written programs in:

BASIC _____ Pascal _____ PL/1 _____ Logo _____ Cobol _____
Others _____ (specify please: _____)

3. Please briefly list any computer work experience, you have had:

1) _____
2) _____

4. Please list all the college mathematics courses you have had:

1) _____ 3) _____ 4) _____
2) _____ 4) _____ 5) _____

5. Do you have access to a computer outside of the university? Yes or No

6. Please place a check beside your current college GPA:

4.0 to 3.5 _____ 2.49 to 2.0 _____
3.49 to 3.0 _____ 1.99 to 1.5 _____
2.99 to 2.5 _____ Below 1.5 _____

7. How would you describe the way you currently feel about computers?

____ Very Nervous
____ Somewhat Nervous
____ Not really nervous, but not really confident either
____ Somewhat Confident
____ Very Confident

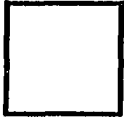
8. Place a check by the computer software packages you have used before:

____ Appleworks _____ Logo _____ Lotus 123
____ Bank Street Writer _____ LogoWriter _____ MacWrite
____ Bank Street Filer _____ SuperPilot _____ MacPaint

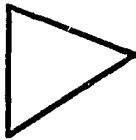
9. Please place a check by the computers you have used before:

____ Apple _____ IBM _____ Zenith _____ Commodore _____ Macintosh
____ Mainframes _____ Other: (Please Specify: _____)

APPENDIX B: EXPERIMENTAL LARGE GROUP ACTIVITY SHEETS

Analogical Reasoning Sheet**Graphic**
(previous graphic output)**Code**
(previously designed code)Sheet # LA2

To Square
Repeat 4 [Fd 50 Rt 90]
End

Graphic
(graphic output desired)**Code**
(modified code needed)**Directions:**

- 1) Sketch the graphical output desired. (*defining the problem*)
- 2) Let's look at a previous problem to help us. (*choosing a plan*)
- 3) Now employ our steps for analogical reasoning. (*carrying out the plan*)

Encode: Writing a few notes, analyze and breakdown the previous graphic output, the desired graphic output, and each procedure in the previous code.

Infer: Step carefully through the previous code, (top right hand corner) to see how it produces the previous graphic (left hand corner).

Map: Draw vertical lines, or describe, the similarities and differences between the previous graphic output and the desired graphic output.

Apply: Now, using what you can from the previous problem, write a program to produce the desired graphical output.

- 4) How did the program work? Describe briefly below: (*looking back*)

Desired Output?
Yes No Not Quite

What Seems Wrong?
describe in words

Part to Modify:
describe in words

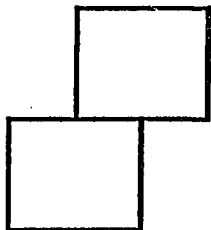
No sweat!
I'll Just Try
Again!

Analogical Reasoning Sheet

Graphic
(previous graphic output)

Code
(previously designed code)

Sheet # LΛ3

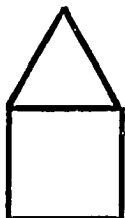


```
To Stack      To Move
Square        Fd 50
Move          Rt 90
Square        Fd 25
End           Lt 90
              End
```

```
To Square
Repeat 4 [Fd 50 Rt 90]
End
```

Graphic
(graphic output desired)

Code
(modified code needed)



Directions:

- 1) Sketch the graphical output desired. (*defining the problem*)
- 2) Let's look at a previous problem to help us. (*choosing a plan*)
- 3) Now employ our steps for analogical reasoning. (*carrying out the plan*)

Encode: Writing a few notes, analyze and breakdown the previous graphic output, the desired graphic output, and each procedure in the previous code.

Infer: Step carefully through the previous code, (top right hand corner) to see how it produces the previous graphic (left hand corner).

Map: Draw vertical lines, or describe, the similarities and differences between the previous graphic output and the desired graphic output.

Apply: Now, using what you can from the previous problem, write a program to produce the desired graphical output.

- 4) How did the program work? Describe briefly below: (*looking back*)

Desired Output?
Yes No Not Quite

What Seems Wrong?
describe in words

Part to Modify:
describe in words

**No sweat!
I'll Just Try
Again!**

Analogical Reasoning Sheet

Graphic
(previous graphic output)



Code
(previously designed code)

Sheet # LA4

To House
Square
Move
Triangle
End

To Square
Repeat 4 [Fd 50 Rt 90]
End

To Move
Fd 50
Rt 30
End

To Triangle
Repeat 3 [Fd 50 Rt 120]
End

(to run type: House)

Graphic
(graphic output desired)



(any size desired)

Code
(modified code needed)

Directions:

- 1) Sketch the graphical output desired. (*defining the problem*)
- 2) Let's look at a previous problem to help us. (*choosing a plan*)
- 3) Now employ our steps for analogical reasoning. (*carrying out the plan*)

Encode: Writing a few notes, analyze and breakdown the previous graphic output, the desired graphic output, and each procedure in the previous code.

Infer: Step carefully through the previous code, (top right hand corner) to see how it produces the previous graphic (left hand corner).

Map: Draw vertical lines, or describe, the similarities and differences between the previous graphic output and the desired graphic output.

Apply: Now, using what you can from the previous problem, write a program to produce the desired graphical output.

- 4) How did the program work? Describe briefly below: (*looking back*)

Desired Output?
Yes No Not Quite

What Seems Wrong?
describe in words

Part to Modify:
describe in words

**No sweat!
I'll Just Try
Again!**

Analogical Reasoning Sheet**Graphic**
*previous graphic output***Code**
*(previously designed code)*Sheet # LA5(filled with orange, any size)
Immediate Mode

To Rectangle :x :y
 Repeat 2 [Fd :x Rt 90 Fd :y Rt 90]
 End

Some commands used to
fill in the immediate mode:

Pu
 Rt 45
 Fd 10
 Pd
 Setc 4
 Fill

Graphic
*(graphic output desired)***Code**
(modified code needed)(any color, and any size)
Programming Mode
(two variable input)**Directions:**

- 1) Sketch the graphical output desired. (*defining the problem*)
- 2) Let's look at a previous problem to help us. (*choosing a plan*)
- 3) Now employ our steps for analogical reasoning. (*carrying out the plan*)

Encode: Writing a few notes, analyze and breakdown the previous graphic output, the desired graphic output, and each procedure in the previous code.**Infer:** Step carefully through the previous code, (top right hand corner) to see how it produces the previous graphic (left hand corner).**Map:** Draw vertical lines, or describe, the similarities and differences between the previous graphic output and the desired graphic output.**Apply:** Now, using what you can from the previous problem, write a program to produce the desired graphical output.

- 4) How did the program work? Describe briefly below: (*looking back*)

Desired Output?
Yes No Not Quite**What Seems Wrong?**
describe in words**Part to Modify:**
describe in words

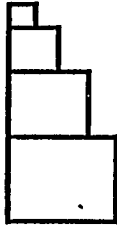
No sweat!
 I'll Just Try
 Again!

Analogical Reasoning Sheet

Graphic
(previous graphic output)

Code
(previously designed code)

Sheet # LA6



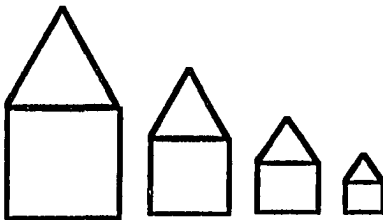
(using recursion)

```
To Stack :X
If :X < 0 [Stop]
Square :X
Move :X
Stack :X - 10
End
```

```
To Move :X
Fd :X
End
To Square :x
Repeat 4 [Fd :x Rt 90]
End
```

Graphic
(graphic output desired)

Code
(modified code needed)



(a recursive program that draws progressively smaller houses)

Directions:

- 1) Sketch the graphical output desired. (*defining the problem*)
- 2) Let's look at a previous problem to help us. (*choosing a plan*)
- 3) Now employ our steps for analogical reasoning. (*carrying out the plan*)
 - Encode:** Writing a few notes, analyze and breakdown the previous graphic output, the desired graphic output, and each procedure in the previous code.
 - Infer:** Step carefully through the previous code, (top right hand corner) to see how it produces the previous graphic (left hand corner).
 - Map:** Draw vertical lines, or describe, the similarities and differences between the previous graphic output and the desired graphic output.
 - Apply:** Now, using what you can from the previous problem, write a program to produce the desired graphical output.

4) How did the program work? Describe briefly below: (*looking back*)

Desired Output?
Yes No Not Quite

What Seems Wrong?
describe in words

Part to Modify:
describe in words

**No sweat!
I'll Just Try
Again!**

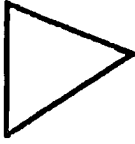
APPENDIX C: CONTROL LARGE GROUP ACTIVITY SHEETS

Class Activity Sheet

Graphic
(graphic output desired)

Code
(code to draw graphic output)

Sheet # LB2

**Directions:**

- 1) Sketch or look at the graphical output desired. (*defining the problem*)
- 2) Think about what you will need to do to have the turtle draw the graphic shown above. You may find it helpful to look at some past problems from your notes. (*choosing a plan*)
- 3) Now try and build a program to have the turtle draw the desired graphic output. You may want to either write out your code in pencil first, or start programming directly on the computer. (*carrying out the plan*)
- 4) How did the program work? Describe briefly below: (*looking back*)

Desired Output?
Yes No Not Quite

What Seems Wrong?
describe in words

Part to Modify:
describe in words

No sweat!
I'll Just Try
Again!

Let's talk about this as a class. What programs did we come up with?

Class Activity Sheet

Graphic
(graphic output desired)

Code
(code to draw graphic output)

Sheet #LB3

Directions:

- 1) Sketch or look at the graphical output desired. (*defining the problem*)
- 2) Think about what you will need to do to have the turtle draw the graphic shown above. You may find it helpful to look at some past problems from your notes. (*choosing a plan*)
- 3) Now try and build a program to have the turtle draw the desired graphic output. You may want to either write out your code in pencil first, or start programming directly on the computer. (*carrying out the plan*)
- 4) How did the program work? Describe briefly below: (*looking back*)

Desired Output?

Yes No Not Quite

What Seems Wrong?

describe in words

Part to Modify:

describe in words

No sweat!

I'll Just Try

Again!

Let's talk about this as a class. What programs did we come up with?

Class Activity Sheet**Graphic**
(graphic output desired)**Code**
(code to draw graphic output)

Sheet #LB4



(any size desired)

Directions:

- 1) Sketch or look at the graphical output desired. (*defining the problem*)
- 2) Think about what you will need to do to have the turtle draw the graphic shown above. You may find it helpful to look at some past problems from your notes. (*choosing a plan*)
- 3) Now try and build a program to have the turtle draw the desired graphic output. You may want to either write out your code in pencil first, or start programming directly on the computer. (*carrying out the plan*)
- 4) How did the program work? Describe briefly below: (*looking back*)

Desired Output?
Yes No Not Quite

What Seems Wrong?
describe in words

Part to Modify:
describe in words

No sweat!
I'll Just Try
Again!

Let's talk about this as a class. What programs did we come up with?

Class Activity Sheet

Graphic
(graphic output desired)

Code
(code to draw graphic output),

Sheet # LB5



(any color, and any size)

Programming Mode
(two variable input)

Directions:

- 1) Sketch or look at the graphical output desired. (*defining the problem*)
- 2) Think about what you will need to do to have the turtle draw the graphic shown above. You may find it helpful to look at some past problems from your notes. (*choosing a plan*)
- 3) Now try and build a program to have the turtle draw the desired graphic output. You may want to either write out your code in pencil first, or start programming directly on the computer. (*carrying out the plan*)
- 4) How did the program work? Describe briefly below: (*looking back*)

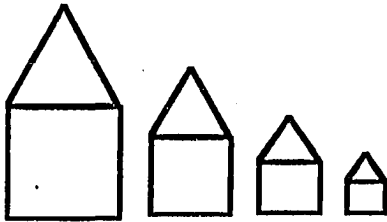
Desired Output?
Yes No Not Quite

What Seems Wrong?
describe in words

Part to Modify:
describe in words

No sweat!
I'll Just Try
Again!

Let's talk about this as a class. What programs did we come up with?

Class Activity SheetGraphic*(graphic output desired)*Code*(code to draw graphic output)*Sheet # LB6

(a recursive program that draws progressively smaller houses)

Directions:

- 1) Sketch or look at the graphical output desired. *(defining the problem)*
- 2) Think about what you will need to do to have the turtle draw the graphic shown above. You may find it helpful to look at some past problems from your notes. *(choosing a plan)*
- 3) Now try and build a program to have the turtle draw the desired graphic output. You may want to either write out your code in pencil first, or start programming directly on the computer. *(carrying out the plan)*
- 4) How did the program work? Describe briefly below: *(looking back)*

Desired Output?

Yes No Not Quite

What Seems Wrong?

describe in words

Part to Modify:

describe in words

No sweat!I'll Just Try
Again!

Let's talk about this as a class. What programs did we come up with?

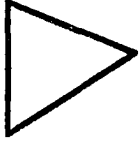
APPENDIX D: EXPERIMENTAL LAB ACTIVITY SHEETS

Analogical Reasoning Sheet

Graphic
(previous graphic output)

Code
(previously designed code)

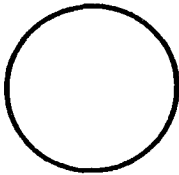
Sheet # SA1



To Triangle
Repeat 3 [Fd 50 Rt 120]
End

Graphic
(graphic output desired)

Code
(modified code needed)



Directions:

- 1) Sketch the graphical output desired. (*defining the problem*)
- 2) Let's look at a previous problem to help us. (*choosing a plan*)
- 3) Now employ our steps for analogical reasoning. (*carrying out the plan*)
 - Encode:** Writing a few notes, analyze and breakdown the previous graphic output, the desired graphic output, and each procedure in the previous code.
 - Infer:** Step carefully through the previous code, (top right hand corner) to see how it produces the previous graphic (left hand corner).
 - Map:** Draw vertical lines, or describe, the similarities and differences between the previous graphic output and the desired graphic output.
 - Apply:** Now, using what you can from the previous problem, write a program to produce the desired graphical output.
- 4) How did the program work? Describe briefly below: (*looking back*)

Desired Output?
Yes No Not Quite

What Seems Wrong?
describe in words

Part to Modify:
describe in words

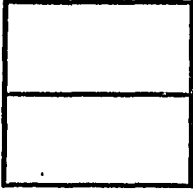
No sweat!
I'll Just Try
Again!

Analogical Reasoning Sheet

Graphic
(previous graphic output)

Code
(previously designed code)

Sheet # SA2



To Stack
Rectangle
Move
Rectangle
End

To Rectangle
Repeat 2 [Fd 25 Rt 90 Fd 50 Rt 90]
End

To Move
Fd 25
End

Graphic
(graphic output desired)

Code
(modified code needed)



Directions:

- 1) Sketch the graphical output desired. (*defining the problem*)
- 2) Let's look at a previous problem to help us. (*choosing a plan*)
- 3) Now employ our steps for analogical reasoning. (*carrying out the plan*)
 - Encode:** Writing a few notes, analyze and breakdown the previous graphic output, the desired graphic output, and each procedure in the previous code.
 - Infer:** Step carefully through the previous code, (top right hand corner) to see how it produces the previous graphic (left hand corner).
 - Map:** Draw vertical lines, or describe, the similarities and differences between the previous graphic output and the desired graphic output.
 - Apply:** Now, using what you can from the previous problem, write a program to produce the desired graphical output.
- 4) How did the program work? Describe briefly below: (*looking back*)

Desired Output?
Yes No Not Quite

What Seems Wrong?
describe in words

Part to Modify:
describe in words

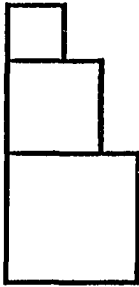
**No sweat!
I'll Just Try
Again!**

Analogical Reasoning Sheet

Graphic
(previous graphic output)

Code
(previously designed code)

Sheet # SA3

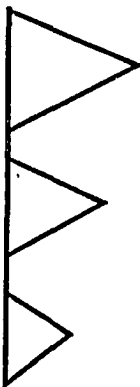


```
To Stack :X
Square :X
Fd :X
Square :X-10
Fd :X-10
Square :X-20
End
```

```
To Square :X
Repeat 4 [Fd :X Rt 90]
End
```

Graphic
(graphic output desired)

Code
(modified code needed)



Directions:

- 1) Sketch the graphical output desired. (*defining the problem*)
- 2) Let's look at a previous problem to help us. (*choosing a plan*)
- 3) Now employ our steps for analogical reasoning. (*carrying out the plan*)

- Encode:** Writing a few notes, analyze and breakdown the previous graphic output, the desired graphic output, and each procedure in the previous code.
- Infer:** Step carefully through the previous code, (top right hand corner) to see how it produces the previous graphic (left hand corner).
- Map:** Draw vertical lines, or describe, the similarities and differences between the previous graphic output and the desired graphic output.
- Apply:** Now, using what you can from the previous problem, write a program to produce the desired graphical output.

- 4) How did the program work? Describe briefly below: (*looking back*)

Desired Output?
Yes No Not Quite

What Seems Wrong?
describe in words

Part to Modify:
describe in words

**No sweat!
I'll Just Try
Again!**

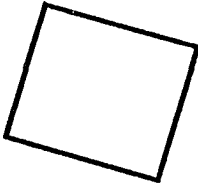
Analogical Reasoning Sheet**Graphic**
(previous graphic output)**Code**
(previously designed code)

Sheet #SA4



To Rectangle :W :L
 Repeat 2 [Fd :W Rt 90 Fd :L Rt 90]
 End

(rectangle with any size of
 length and width upon input)

Graphic
(graphic output desired)**Code**
(modified code needed)

(a square of any size tilted
 at any angle)

Directions:

- 1) Sketch the graphical output desired. (*defining the problem*)
- 2) Let's look at a previous problem to help us. (*choosing a plan*)
- 3) Now employ our steps for analogical reasoning. (*carrying out the plan*)

Encode: Writing a few notes, analyze and breakdown the previous graphic output, the desired graphic output, and each procedure in the previous code.

Infer: Step carefully through the previous code, (top right hand corner) to see how it produces the previous graphic (left hand corner).

Map: Draw vertical lines, or describe, the similarities and differences between the previous graphic output and the desired graphic output.

Apply: Now, using what you can from the previous problem, write a program to produce the desired graphical output.

- 4) How did the program work? Describe briefly below: (*looking back*)

Desired Output?
 Yes No Not Quite

What Seems Wrong?
 describe in words

Part to Modify:
 describe in words

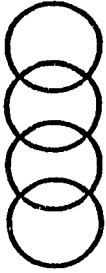
**No sweat!
 I'll Just Try
 Again!**

Analogical Reasoning Sheet

Graphic
(previous graphic output)

Code
(previously designed code)

Sheet # SA5



```
To Coll :X
If :X < 0 [Stop]
Circle
Move
Coll :X-1
End
```

```
To Circle
Repeat 36 [Fd 2 Rt 10]
End

To Move
Pu
Fd 20
Pd
End
```

Graphic
(graphic output desired)

Code
(modified code needed)



Directions:

- 1) Sketch the graphical output desired. (*defining the problem*)
- 2) Let's look at a previous problem to help us. (*choosing a plan*)
- 3) Now employ our steps for analogical reasoning. (*carrying out the plan*)
 - Encode:** Writing a few notes, analyze and breakdown the previous graphic output, the desired graphic output, and each procedure in the previous code.
 - Infer:** Step carefully through the previous code, (top right hand corner) to see how it produces the previous graphic (left hand corner).
 - Map:** Draw vertical lines, or describe, the similarities and differences between the previous graphic output and the desired graphic output.
 - Apply:** Now, using what you can from the previous problem, write a program to produce the desired graphical output.

4) How did the program work? Describe briefly below: (*looking back*)

Desired Output?
Yes No Not Quite

What Seems Wrong?
describe in words

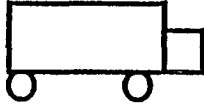
Part to Modify:
describe in words

No sweat!
I'll Just Try
Again!

Group A

Analogical Reasoning Homework #1 Project Planning Sheet

Graphic
(previous graphic output)



Graphic
(graphic output desired)

Code
(previously designed code)

Calling Procedure

To Truck

Body
Move1
Wheel
Move2
Wheel
Move3
Cab
End

Sub-Procedures

To Body

Repeat 2[Fd 20 Rt 90 Fd 50 Rt 90]
End

To Wheel

Repeat 10[Fd 3 Rt 36]
End

To Cab

Repeat 4[Fd 10 Rt 90]
Ht
End

To Move1

Pu
Rt 90
Fd 5
Pd
End

To Move2

Pu
Fd 40
Pd
End

To Move3

Pu
Fd 5
Lt 90
Pd
End

Code
(modified code needed)

Remember to plan your project by using the steps below!

- 1) Sketch the graphical output desired. (*defining the problem*)
- 2) Let's look at previous problems to help us. (*choosing a plan*)

My project will also use other sheets: Sheet # LA1 Part: the square
 (as well as this sheet) Sheet # _____ Part: _____
 Sheet # _____ Part: _____
 Sheet # _____ Part: _____

- 3) Now employ the steps for analogical reasoning. (*carrying out the plan*)

Encode: Look carefully at the sketch of your project, and the graphic and program for the truck.
 (*try to break these into smaller parts*)

Infer: Now step through the program for the truck and see how it draws the truck

Map: Now look at how parts of your graphic are similar to, and different from, parts of the truck.

Apply: Now try and pencil out a program for your project. You will probably need to repeat the encode, infer, and map steps occasionally. You will also want to look carefully at parts of other problems you use, in the same way.

- 4) How did your program work? You may want to keep a record as you try things. (*looking back*)

LogoWriter Lab Assignment #1

The first LogoWriter assignment is to create a program which draws a simple graphic picture, similar to the truck example distributed in class. The program should use at least 7 separate procedures, and be executed by typing the name of a single calling procedure. It should be stored on your LogoWriter disk within a page called Lastname1. (for instance Smith1).

CRITERIA

At least 6 separate procedures used in the program.....	4	_____
Project runs without errors.....	3	_____
Completed <i>Homework Project Planning Sheet</i> turned in	2	_____
Project has a theme.....	2	_____
Project executes by the typing of a single procedure name.....	2	_____
Project is saved correctly, (see below).....	1	_____
Total.....	14	_____

Project saved under a page named: _____ (Lastname1)
to run the project type: _____ (name of the calling procedure)

Some reminders about saving your project:

Naming a LogoWriter page:

Remember, the LogoWriter "page", where your program is to be stored must first be named with the NP command before your program can be saved. This naming of a page does not save your work, but only sets up LogoWriter so that your work will be automatically saved when you press the escape key.

To name a page for LogoWriter project #1 type:
NP "lastname1 (and press return)

Thus to save your project while you are working:

- 1) Check the top of the screen to verify that the page you are working on already has a name. If it doesn't, name it with the NP command.
- 2) Press escape, this will bring you back to the contents page, where the page name should now be listed.

To turn in your project to your lab instructor:

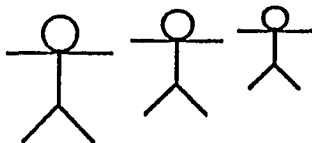
- 1) At the beginning of the lab period, first boot up LogoWriter, and run your project so that it shows on the screen. Make sure that the page is already named with Lastname1 (for instance: Smith1)
- 2) Remove your LogoWriter disk from the drive.
- 3) Place the Lab Instructor's disk in the drive.
- 4) Press escape, your project is now saved on the instructor's disk within a page identified with your last name.

Group A

Analogical Reasoning Homework #2 Project Planning Sheet

Graphic

(previous graphic output)



Calling Procedure

To Men
 Startcorner
 Oneman 40
 Moveover 40
 Oneman 30
 Moveover 30
 Oneman 20
 End

Code

(previously designed code)

Sub-Procedures

To Oneman :X	To Legs :X	To Body :X
Legs :X	Rt 45	Fd :X
Body :X	Fd :X	End
Head :X	Rt 90	
Arms :X	Fd :X	To Arms :X
End	Bk :X	Lt 90
	Lt 45	Fd :X
To Head :X	Lt 90	Rt 180
Lt 90	End	Fd :X * 2
Repeat 36 [Fd :X/20 Rt 10]		Lt 90
End		End

Graphic

(graphic output desired)

Code

(modified code needed)

Remember to plan your project by using the steps below!

- 1) Sketch the graphical output desired. (defining the problem)
- 2) Let's look at previous problems to help us choose a plan

My project will also use other sheets: Sheet # LA1 Part: the square
 Sheet # Part:
 Sheet # Part:
 Sheet # Part:

- 3) Now employ the steps for analogical reasoning. (carrying out the plan)

Encode: Look carefully at the sketch of your project, and the graphic and program for the seascene.

(try to break these into smaller parts)

Infer: Now step through the program for the boats and see how it draws the seascene.

Map: Now look at how parts of your graphic are similar to, and different from, parts of the seascene.

Apply: Now try and pencil out a program for your project. You will probably need to repeat the encode, infer, and map steps occasionally. You will also want to look carefully at parts of other problems you use, in the same way.

- 4) How did your program work? You may want to keep a record as you try things. (looking back)

LogoWriter Lab Assignment #2

The second LogoWriter assignment is to create a program using variables which draws a simple graphic picture, similar to the stickmen example distributed in class. The program should use at least 7 separate procedures, and be executed by typing the name of a single calling procedure. The project should also use variables somewhere in the program. It should be stored on your LogoWriter disk within a page called Lastname2, (for instance Smith2).

CRITERIA

At least 6 separate procedures used in the program.....	2	_____
Project runs without errors.....	2	_____
Project uses variables.....	4	_____
Completed <i>Homework Project Planning Sheet</i> turned in	2	_____
Project has a theme.....	2	_____
Project executes by the typing of a single procedure name.....	1	_____
Project is saved correctly, (see below).....	1	_____
Total.....	14	_____

Project saved under a page named: _____ (Lastname1)
 To run the project type: _____ (name of the calling procedure)

Some reminders about saving your project:

Naming a LogoWriter page:

Remember, the LogoWriter "page", where your program is to be stored must first be named with the NP command before your program can be saved. This naming of a page does not save your work, but only sets up LogoWriter so that your work will be automatically saved when you press the escape key.

To name a page for LogoWriter project #1 type:
 NP "lastname1 (and press return)

Thus to save your project while you are working:

- 1) Check the top of the screen to verify that the page you are working on already has a name. If it doesn't, name it with the NP command.
- 2) Press escape, this will bring you back to the contents page, where the page name should now be listed.

To turn in your project to your lab instructor:

- 1) At the beginning of the lab period, first boot up LogoWriter, and run your project so that it shows on the screen. Make sure that the page is already named with Lastname1 (for instance: Smith1)
- 2) Remove your LogoWriter disk from the drive.
- 3) Place the Lab Instructor's disk in the drive.
- 4) Press escape, your project is now saved on the instructors disk within a page identified with your last name.

Group A Analogical Reasoning Homework #3 Project Planning Sheet

Graphic
(previous graphic output)



Graphic
(graphic output desired)

Code
(previously designed code)

<p>Calling Procedure</p> <p>To SeaScene Waves 10 Moveleft Boat 40 Move 40 Boat 30 Move 30 Boat 20 End</p>	<p>Sub-Procedures</p> <p>To Waves :x If :x < 0 [stop] Onewave Waves :x-1 End</p> <p>To Onewave Repeat 180 [Fd .1 Rt 1] Lt 180 End</p> <p>To Boat :y Body :y Mast :y End</p>	<p>Code (modified code needed)</p> <p>To Body :y Repeat 2 [Fd :y/2 Rt 90 Fd :y Rt 90] End</p> <p>To Mast :y Fd :y/2 Rt 90 Fd :y/2 Lt 90 Fd :y/2 Repeat 3 [Fd :y/2 Rt 120] End</p>	<p>Code</p> <p>To Move :y Pu Bk :y Rt 90 Fd :y Lt 90 Pd End</p> <p>To Moveleft Pu Home Fd 10 Pd End</p>
--	---	--	--

Remember to plan your project by using the steps below!

- 1) Sketch the graphical output desired. *(defining the problem)*
- 2) Let's look at previous problems to help us *(choosing a plan)*
 My project will also use other sheets: Sheet # LAI Part: the square
 (as well as this sheet) Sheet # _____ Part: _____
 Sheet # _____ Part: _____
 Sheet # _____ Part: _____
- 3) Now employ the steps for analogical reasoning. *(carrying out the plan)*
 - Encode:** Look carefully at the sketch of your project, and the graphic and program for the seascape. *(try to break these into smaller parts)*
 - Infer:** Now step through the program for the boats and see how it draws the seascape.
 - Map:** Now look at how parts of your graphic are similar to, and different from, parts of the seascape.
 - Apply:** Now try and pencil out a program for your project. You will probably need to repeat the encode, infer, and map steps occasionally. You will also want to look carefully at parts of other problems you use, in the same way.
- 4) How did your program work? You may want to keep a record as you try things. *(looking back)*

LogoWriter Lab Assignment #3

The third LogoWriter assignment is to create a program using recursion which draws a simple graphic picture, similar to the SeaScene example distributed in class. The program should use at least 7 separate procedures, and be executed by typing the name of a single calling procedure. The project should also use recursion somewhere in the program. It should be stored on your LogoWriter disk within a page called Lastname3, (for instance Smith3).

CRITERIA

At least 6 separate procedures used in the program.....	2	_____
Project runs without errors.....	2	_____
Project uses recursion.....	4	_____
Completed <i>Homework Project Planning Sheet</i> turned in	2	_____
Project has a theme.....	2	_____
Project executes by the typing of a single procedure name.....	1	_____
Project is saved correctly, (see below).....	1	_____
Total.....	14	_____

Project saved under a page named: _____ (Lastname1)
 To run the project type: _____ (name of the calling procedure)

Some reminders about saving your project:

Naming a LogoWriter page:

Remember, the LogoWriter "page", where your program is to be stored must first be named with the NP command before your program can be saved. This naming of a page does not save your work, but only sets up LogoWriter so that your work will be automatically saved when you press the escape key.

To name a page for LogoWriter project #1 type:
 NP "lastname1 (and press return)

Thus to save your project while you are working:

- 1) Check the top of the screen to verify that the page you are working on already has a name. If it doesn't, name it with the NP command.
- 2) Press escape, this will bring you back to the contents page, where the page name should now be listed.

To turn in your project to your lab instructor:

- 1) At the beginning of the lab period, first boot up LogoWriter, and run your project so that it shows on the screen. Make sure that the page is already named with Lastname1 (for instance: Smith1)
- 2) Remove your LogoWriter disk from the drive.
- 3) Place the Lab Instructor's disk in the drive.
- 4) Press escape, your project is now saved on the instructor's disk within a page identified with your last name.

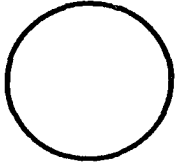
APPENDIX E: CONTROL LAB ACTIVITY SHEETS

Class Activity Sheet

Graphic
(graphic output desired)

Code
(code to draw graphic output)

Sheet # SB1

Directions:

- 1) Sketch or look at the graphical output desired. (*defining the problem*)
- 2) Think about what you will need to do to have the turtle draw the graphic shown above. You may find it helpful to look at some past problems from your notes. (*choosing a plan*)
- 3) Now try and build a program to have the turtle draw the desired graphic output. You may want to either write out your code in pencil first, or start programming directly on the computer. (*carrying out the plan*)
- 4) How did the program work? Describe briefly below: (*looking back*)

Desired Output?
Yes No Not Quite

What Seems Wrong?
describe in words

Part to Modify:
describe in words

No sweat!
I'll Just Try
Again!

Let's talk about this as a class. What programs did we come up with?

Class Activity Sheet**Graphic**
(graphic output desired)**Code**
(code to draw graphic output)Sheet # SB2**Directions:**

- 1) Sketch or look at the graphical output desired. (*defining the problem*)
- 2) Think about what you will need to do to have the turtle draw the graphic shown above. You may find it helpful to look at some past problems from your notes. (*choosing a plan*)
- 3) Now try and build a program to have the turtle draw the desired graphic output. You may want to either write out your code in pencil first, or start programming directly on the computer. (*carrying out the plan*)
- 4) How did the program work? Describe briefly below: (*looking back*)

Desired Output?
Yes No Not Quite

What Seems Wrong?
describe in words

Part to Modify:
describe in words

No sweat!
I'll Just Try
Again!

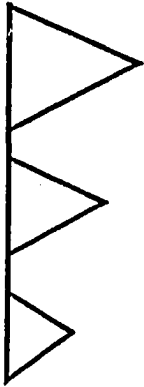
Let's talk about this as a class. What programs did we come up with?

Class Activity Sheet

Graphic
(graphic output desired)

Code
(code to draw graphic output)

Sheet # SB3

**Directions:**

- 1) Sketch or look at the graphical output desired. (*defining the problem*)
- 2) Think about what you will need to do to have the turtle draw the graphic shown above. You may find it helpful to look at some past problems from your notes. (*choosing a plan*)
- 3) Now try and build a program to have the turtle draw the desired graphic output. You may want to either write out your code in pencil first, or start programming directly on the computer. (*carrying out the plan*)
- 4) How did the program work? Describe briefly below: (*looking back*)

Desired Output?
Yes No Not Quite

What Seems Wrong?
describe in words

Part to Modify:
describe in words

No sweat!
I'll Just Try
Again!

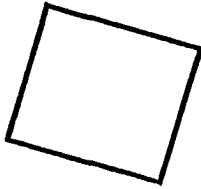
Let's talk about this as a class. What programs did we come up with?

Class Activity Sheet

Graphic
(graphic output desired)

Code
(code to draw graphic output)

Sheet #SB4



(a square of any size tilted
at any angle)

Directions:

- 1) Sketch or look at the graphical output desired. (*defining the problem*)
- 2) Think about what you will need to do to have the turtle draw the graphic shown above. You may find it helpful to look at some past problems from your notes. (*choosing a plan*)
- 3) Now try and build a program to have the turtle draw the desired graphic output. You may want to either write out your code in pencil first, or start programming directly on the computer. (*carrying out the plan*)
- 4) How did the program work? Describe briefly below: (*looking back*)

Desired Output?

Yes No Not Quite

What Seems Wrong?

describe in words

Part to Modify:

describe in words

No sweat!

I'll Just Try

Again!

Let's talk about this as a class. What programs did we come up with?

Class Activity Sheet

Graphic
(graphic output desired)

Code
(code to draw graphic output)

Sheet #SB5

Directions:

- 1) Sketch or look at the graphical output desired. (*defining the problem*)
- 2) Think about what you will need to do to have the turtle draw the graphic shown above. You may find it helpful to look at some past problems from your notes. (*choosing a plan*)
- 3) Now try and build a program to have the turtle draw the desired graphic output. You may want to either write out your code in pencil first, or start programming directly on the computer. (*carrying out the plan*)
- 4) How did the program work? Describe briefly below: (*looking back*)

Desired Output?
Yes No Not Quite

What Seems Wrong?
describe in words

Part to Modify:
describe in words

No sweat!
I'll Just Try
Again!

Let's talk about this as a class. What programs did we come up with?

Group B

Class Activity Homework #1 Project Planning Sheet

Graphic

(graphic output desired)

Code

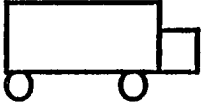
(program to draw graphic output desired)

Remember to plan your project by using the steps below!

- 1) Sketch or look at the graphical output desired. (*defining the problem*)
- 2) Think about what you will need to do to have the turtle draw the graphic you have sketched above. You may find it helpful to look at some past problems from your notes. (*choosing a plan*)
- 3) Now try and build a program to have the turtle draw the your desired graphic output. You may want to either write out your code in pencil first, or start programming directly on the computer. (*carrying out the plan*)
Eventually you will need to turn this sheet in with your program written out.
- 4) How did the program work? You may want to keep a record. (*looking back*)

Yes No Not Quite describe in words describe in words

Let's talk about this as a class. What programs did we come up with?

Homework Example #1Calling Procedure

To Truck
Body
Move1
Wheel
Move2
Wheel
Move3
Cab
End

Sub-Procedures

To Body
Repeat 2[Fd 20 Rt 90 Fd 50 Rt 90]
End

To Wheel
Repeat 10[Fd 3 Rt 36]
End

To Cab
Repeat 4[Fd 10 Rt 90]
Ht
End

To Move1

Pu
Rt 90
Fd 5
Pd
End

To Move2

Pu
Fd 40
Pd
End

To Move3

Pu
Fd 5
Lt 90
Pd
End

LogoWriter Lab Assignment #1

The first LogoWriter assignment is to create a program which draws a simple graphic picture, similar to the truck example distributed in class. The program should use at least 7 separate procedures, and be executed by typing the name of a single calling procedure. It should be stored on your LogoWriter disk within a page called Lastname1, (for instance Smith1).

CRITERIA

At least 6 separate procedures used in the program.....	4	_____
Project runs without errors.....	3	_____
Completed <i>Homework Project Planning Sheet</i> turned in	2	_____
Project has a theme.....	2	_____
Project executes by the typing of a single procedure name.....	2	_____
Project is saved correctly, (see below),.....	1	_____
Total.....	14	_____

Project saved under a page named: _____ (Lastname1)
 To run the project type: _____ (name of the calling procedure)

Some reminders about saving your project:

Naming a LogoWriter page:

Remember, the LogoWriter "page", where your program is to be stored must first be named with the NP command before your program can be saved. This naming of a page does not save your work, but only sets up LogoWriter so that your work will be automatically saved when you press the escape key.

To name a page for LogoWriter project #1 type:
 NP "lastname1 (and press return)

Thus to save your project while you are working:

- 1) Check the top of the screen to verify that the page you are working on already has a name. If it doesn't, name it with the NP command.
- 2) Press escape, this will bring you back to the contents page, where the page name should now be listed.

To turn in your project to your lab instructor:

- 1) At the beginning of the lab period, first boot up LogoWriter, and run your project so that it shows on the screen. Make sure that the page is already named with Lastname1 (for instance: Smith1)
- 2) Remove your LogoWriter disk from the drive.
- 3) Place the Lab Instructor's disk in the drive.
- 4) Press escape, your project is now saved on the Instructor's disk within a page identified with your last name.

Group B

Class Activity Homework #2 Project Planning Sheet

Graphic
(graphic output desired)

Code
(program to draw graphic output desired)

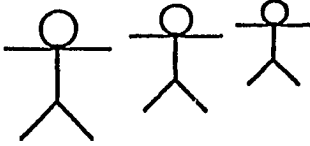
Remember to plan your project by using the steps below!

- 1) Sketch or look at the graphical output desired. (*defining the problem*)
- 2) Think about what you will need to do to have the turtle draw the graphic you have sketched above. You may find it helpful to look at some past problems from your notes. (*choosing a plan*)
- 3) Now try and build a program to have the turtle draw the your desired graphic output. You may want to either write out your code in pencil first, or start programming directly on the computer. (*carrying out the plan*)
Eventually you will need to turn this sheet in with your program written out.
- 4) How did the program work? You may want to keep a record. (*looking back*)

Yes No Not Quite describe in words describe in words

Let's talk about this as a class. What programs did we come up with?

Graphic
(previous graphic output)



Calling Procedure

To Men
Startcorner
Oneman 40
Moveover 40
Oneman 30
Moveover 30
Oneman 20
End

Code
(previously designed code)

Sub-Procedures

To Oneman :X	To Legs :X	To Body :X
Legs :X	Rt 45	Fd :X
Body :X	Fd :X	End
Head :X	Rt 90	To Arms :X
Arms :X	Fd :X	Lt 90
End	Blk :X	Fd :X
	Lt 45	Rt 180
To Head :X	Lt 90	Fd :X * 2
Lt 90	End	Lt 90
Repeat 36 [Fd :X/20 Rt 10]		End
End		

LogoWriter Lab Assignment #2

The second LogoWriter assignment is to create a program using variables which draws a simple graphic picture, similar to the stickmen example distributed in class. The program should use at least 7 separate procedures, and be executed by typing the name of a single calling procedure. The project should also use variables somewhere in the program. It should be stored on your LogoWriter disk within a page called Lastname2, (for instance Smith2).

CRITERIA

At least 6 separate procedures used in the program.....	2	_____
Project runs without errors.....	2	_____
Project uses variables.....	4	_____
Completed <i>Homework Project Planning Sheet</i> turned in	2	_____
Project has a theme.....	2	_____
Project executes by the typing of a single procedure name.....	1	_____
Project is saved correctly, (see below),.....	1	_____
Total.....	14	_____

Project saved under a page named: _____ (Lastname1)
 To run the project type: _____ (name of the calling procedure)

Some reminders about saving your project:**Naming a LogoWriter page:**

Remember, the LogoWriter "page", where your program is to be stored must first be named with the NP command before your program can be saved. This naming of a page does not save your work, but only sets up LogoWriter so that your work will be automatically saved when you press the escape key.

To name a page for LogoWriter project #1 type:
 NP "lastname1 (and press return)

Thus to save your project while you are working:

- 1) Check the top of the screen to verify that the page you are working on already has a name. If it doesn't, name it with the NP command.
- 2) Press escape, this will bring you back to the contents page, where the page name should now be listed.

To turn in your project to your lab instructor:

- 1) At the beginning of the lab period, first boot up LogoWriter, and run your project so that it shows on the screen. Make sure that the page is already named with Lastname1 (for instance: Smith1)
- 2) Remove your LogoWriter disk from the drive.
- 3) Place the Lab Instructor's disk in the drive.
- 4) Press escape, your project is now saved on the instructors disk within a page identified with your last name.

Group B

Class Activity Homework #3 Project Planning Sheet

Graphic

(graphic output desired)

Code

(program to draw graphic output desired)

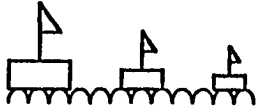
Remember to plan your project by using the steps below!

- 1) Sketch or look at the graphical output desired. (*defining the problem*)
- 2) Think about what you will need to do to have the turtle draw the graphic you have sketched above. You may find it helpful to look at some past problems from your notes. (*choosing a plan*)
- 3) Now try and build a program to have the turtle draw the your desired graphic output. You may want to either write out your code in pencil first, or start programming directly on the computer. (*carrying out the plan*)
Eventually you will need to turn this sheet in with your program written out.
- 4) How did the program work? You may want to keep a record. (*looking back*)

Yes No Not Quite describe in words describe in words

Let's talk about this as a class. What programs did we come up with?

Graphic
(previous graphic output)



Calling Procedure

```
To SeaScene
Waves 10
Moveleft
Boat 40
Move 40
Boat 30
Move 30
Boat 20
End

To Waves :x
If :x < 0 [stop]
Onewave
Waves :x-1
End

To Onewave
Repeat 180 [Fd .1 Rt 1]
Lt 180
End

To Boat :y
Body :y
Mast :y
End
```

Code
(previously designed code)

Sub-Procedures

```
To Body :y
Repeat 2 [Fd :y/2 Rt 90 Fd :y Rt 90]
End

To Mast :y
Fd :y/2
Rt 90
Fd :y/2
Lt 90
Fd :y/2
Repeat 3 [Fd :y/2 Rt 120]
End
```


LogoWriter Lab Assignment #3

The third LogoWriter assignment is to create a program using recursion which draws a simple graphic picture, similar to the SeaScene example distributed in class. The program should use at least 7 separate procedures, and be executed by typing the name of a single calling procedure. The project should also use recursion somewhere in the program. It should be stored on your LogoWriter disk within a page called Lastname3, (for instance Smith3).

CRITERIA

At least 6 separate procedures used in the program.....	2	___
Project runs without errors.....	2	___
Project uses recursion.....	4	___
Completed <i>Homework Project Planning Sheet</i> turned in	2	___
Project has a theme.....	2	___
Project executes by the typing of a single procedure name.....	1	___
Project is saved correctly, (see below).....	1	___
Total.....	14	___

Project saved under a page named: _____ (Lastname1)
 To run the project type: _____ (name of the calling procedure)

Some reminders about saving your project:

Naming a LogoWriter page:

Remember, the LogoWriter "page", where your program is to be stored must first be named with the NP command before your program can be saved. This naming of a page does not save your work, but only sets up LogoWriter so that your work will be automatically saved when you press the escape key.

To name a page for LogoWriter project #1 type:
 NP "lastname1" (and press return)

Thus to save your project while you are working:

- 1) Check the top of the screen to verify that the page you are working on already has a name. If it doesn't, name it with the NP command.
- 2) Press escape, this will bring you back to the contents page, where the page name should now be listed.

To turn in your project to your lab instructor:

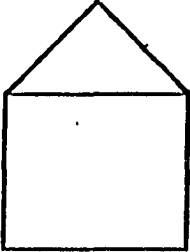
- 1) At the beginning of the lab period, first boot up LogoWriter, and run your project so that it shows on the screen. Make sure that the page is already named with Lastname1 (for instance: Smith1)
- 2) Remove your LogoWriter disk from the drive.
- 3) Place the Lab Instructor's disk in the drive.
- 4) Press escape, your project is now saved on the instructor's disk within a page identified with your last name.

**APPENDIX F: ANALOGICAL REASONING PROGRAMMING
INSTRUCTIONAL TECHNIQUE**

Using Sternberg's Component Model to organize programming instruction for possible analogical reasoning transfer.

Problem done in the past by students:

A



B

```
TO SHACK
  SQUARE
  MOVE1
  TRIANGLE
END
```

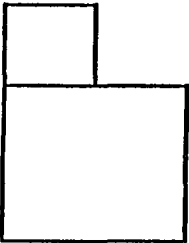
```
TO SQUARE
  REPEAT 4 [FD 50 RT 90]
END
```

```
TO MOVE1
  FD 50
  RT 30
END
```

```
TO TRIANGLE
  REPEAT 3 [FD 50 RT 120]
END
```

Problem to be done by the students

C



D
?

Students would be asked to write code for the graphic to the left.

Instructional process:

Students would have previously developed the code for the SHACK graphic (A). Based on this knowledge, they would now be asked to develop a program to draw figure C.

The experimental group would be formally encouraged to follow the following steps explicitly, while the control group would be merely directed to analyze the previous problem for help in developing the new code.

Instructional Steps:

Encode First students in the experimental group would be directed to look at the SHACK graphic (A) and understand its parts. Then students would be directed to look at the code in the program for shack (B) and try to understand its parts. Finally, the parts of the graphic in C would be looked at. The students would be told they are "Encoding"

Infer The students in the experimental group would now be told to analyze the relationship between the parts of graphic in A and the parts or subprocedures of the code in B. They would be told they are now "Infering".

Map Students in the experimental group would now be told to look at the graphic in A and the graphic in C to analyze the relationship or similarity between the two graphics. They would be told that they are now "mapping".

Apply Students in the experimental group would now be told to "Apply" by generating a possible code (D) to the new graphic in C. They would then test their code, as would the control group, and eventually discuss possible solutions.

**APPENDIX G: EXPERIMENTAL LARGE GROUP INSTRUCTOR
OUTLINES**

232
Lecture Day 1
Group A
General Outline

Attendance:

- 1) Verbally ask students to make sure that they are in the A group by checking their schedules, or if needed, the master lists at the front of the room.
- 2) Pass around the split lecture attendance sheets.

Announcements:

- 1) Remind students that split labs start Wednesday September 28.
- 2) Remind students to quickly purchase LogoWriter Disks in IRC.
- 3) Other: _____

Introduce Logo and the Logo Philosophy

- Transparency- •Introduce Logo and the Logo philosophy.
 •Indicate that we will discuss the skill of analogical reasoning in more detail as a skill typical of programming.

Introduce Analogical Reasoning

- Transparency- •Define analogical reasoning
Transparency- •Discuss examples of analogical reasoning

Illustrate the Process of Analogical Reasoning

- Transparency- •Mention that analogical is a very global skill, but that it has been attempted to be duplicated when people take tests using analogies. Typically, what is involved in such a test is the process shown.

Relate Analogical Reasoning to Programming

- Transparency- •Discuss how expert programmers use analogical reasoning.

Introduce LogoWriter

- Boot up Disk- •Show initial entry screen, choosing new page
 •Show the following primitives:
(asking students to take notes, and predict the outcome of typed primitive commands)

FD xx	RT xx	PU	CG	HT
BK xx	LT xx	PD	HOME	ST

Distribute End of Period Mini-Quiz (if time)

Pass out the quiz, allow students to answer right on the half sheet of paper. Remind students to place name, lab section, and A/B group on quiz.

233
Lecture Day 2
Group A - General Outline

Attendance:

- 1) Verbally ask students to make sure that they are in the A group by checking their schedules, or if needed, the master lists at the front of the room. (*right before class starts*)
- 2) Pass around the split lecture attendance sheets.

Announcements:

- 1) Remind students that split labs start Wednesday September 28.
- 2) Remind students to quickly purchase LogoWriter Disks in IRC.
- 3) Other: _____

Review Primitives

- On Computer-
- Review briefly booting up LogoWriter as it boots.
 - Review some primitives
(emphasize PU, PD, CG, HOME)
 - Discuss primitive sequence for drawing a square

Introduce the Repeat Command

- On Computer-
- Introduce the repeat command with a square:
Repeat 4 [Fd 50 Rt 90]
 - Ask students to predict what happens when a change is made:
Repeat 4 -----> Repeat 8

Introduce Procedures

- On Computer-
- Show how to define procedures by use of a square:
(Using open-apple-f editor)

To Square
Repeat 4 [Fd 50 Rt 90]
End
 - Change square within the editor to a different size.
Change Fd 50 -----> Fd 30
 - Show that the computer now knows a new word by:
Repeat 5 [Square Rt 45]

Activity Sheet L2A- (pass out sheet now)

(Students should always write on these sheets in case of a quiz, which may or may not be open notes!)

- Transparency-
- Lead students in discussion through steps of sheet
(ENCODE, INFER, MAP, APPLY)
 - REMIND students that we are essentially using a

process that is to help us understand a new problem based on what we know from a previous problem

- On Computer-
- Show a student example of the solution (if running out of time use the transparency answer)
 - Mention or discuss the angle of 120 degrees

Namepage Command and Saving

- On Computer-
- Show namepage command of: Np "XXXXXX"
 - Discuss necessity of hitting escape

Discuss Homework Assignment #1

(all three of the following will be passed out and rediscussed in lab)

- Transparency- •Show example project for Homework #1
- project should be as extensive, run with one command, and be broken into parts.
 - students should plan first in pencil by carefully looking at the example project.
 - students should take advantage of previous sheets
- Transparency- •Show the grading criteria sheet for the project.
- students will need to turn in a project on disk, a planning sheet, and a criteria sheet.

Administer End of Period Mini-Quiz (if time)

237
Lecture Day 4
Group A - General Outline

Attendance:

- 1) Ask students to make sure that they are in the A group by checking their schedules, or if needed, the master lists at the front of the room. (*right before class starts*)
- 2) Pass around the split lecture attendance sheets.

Announcements:

- 1) Remind students they should be attending split labs now.
(*Lists at the front of the room will show where students need to go*)
- 2) Remind students to purchase LogoWriter Disks in the IRC before lab.
- 3) Remind students that their first LogoWriter project will be due at the start of their second LogoWriter Lab, and that three things will need to be turned in:
 - 1) a project on disk of at least 6 procedures
 - 2) a planning sheet with a written copy of the program
 - 3) a criteria sheet with the name of the project
- 3) Other announcements: _____

Introduce Variables

On Computer-

•Briefly show the square procedure:

To Square

Repeat 4 [Fd 50 Rt 90]

End

discuss that if we want a square of a different size, we need to go in and actually change the procedure or retype with a slightly different name.

To Square :X

Repeat 4 [Fd :X Rt 90]

End

show that this procedure is much more powerful and flexible.

•Show the Boxes procedure using variables:

To Boxes

Square 50

Square 40

Square 30

End

Type: Boxes

Thus procedures using variables can be placed within other procedures.

•Modify the Boxes procedure to take input:

To Boxes :X

Square :X

Square :X-10

Square :X-20

End

Type: Boxes 50, or Boxes 70, etc...

Thus the variable can be passed from an input to the calling procedure to internal subprocedures.

Activity Sheet #LA4 (pass out sheet now)

- Transparency- •Lead students in discussion of steps on the sheet
(ENCODE, INFER, MAP, APPLY)
•Emphasize how looking back at past problems help
(in this case, the procedure we did for a house helps)
- On Computer- •Show a typical example (either student's or prepared one)
- Transparency- •Debrief the answer shown on the transparency
•Emphasize the syntax format for variable procedures

Mention Homework Assignment #2

- Merely mention that homework assignment number 2 will be similar to the first, but will use variables.
- It will be discussed in more depth next lecture and in lab

Administer End of Period Mini-Quiz (if time)

239
Lecture Day 5
Group A
General Outline

Announcements:

- Remind students that their quizzes will be returned during the second LogoWriter lab.
- Remind students that the Lecture midterm will be Thursday, Oct. 27
- Remind students that Lab Midterms will begin Wednesday, Oct. 19
- Remind students that Thursday will be the last split lecture, but that split labs will continue for a total of three Logowriter lab meetings.
- Other -----

Review Single Variable Procedures

- On Computer- •Show the single variable procedure for a rectangle:
 •Run the procedure with various inputs

```
To Rectangle :W
Repeat 2 [Fd :W Rt 90 Fd 100 Rt 90]
End
```

Introduce Two Variable Procedures

- On Computer- •Modify the rectangle procedure to use two inputs:
 •Run the procedure with various inputs

```
To Rectangle :W :L
Repeat 2 [Fd :W Rt 90 Fd :L Rt 90]
End
```

Show the Fill Command

- On Computer- •Draw a rectangle of typical dimensions
 (use the rectangle procedure in the editor)

- Fill in the rectangle by use of the following:

<u>Type in Immediate Mode:</u>	<u>Colors:(on chalkboard)</u>
Pu	0 Black
Rt 45	1 White
Fd 10	2 Green
Pd	3 Violet
SetC 1	4 Orange
Fill	5 Blue

- Clear the screen, and try again with a different color
- Emphasize that LogoWriter will not fill when the turtle is setting on a line.

Activity Sheet #LA5 (pass out sheet now)

- Transparency- •Lead students through steps on the sheet
 •Ask students to identify what the next step is and what is to be done during that step.
 (ENCODE, INFER, MAP, APPLY)
- On Computer- •Show a typical answer (student or prepared)
 •Emphasize that it might be useful to move back out of the square at the end of the procedure.

Review the General Nature of Analogical Reasoning

- Transparency- •Remind students of the general definition of analogical reasoning, and that we have been attempting to use it in helping us program.
- Transparency- •Show the Coaches Problem
 Transparency
 •Brainstorm other possible applications

Review Homework Assignment #2

- Transparency- •Show "stick people" example project using variables and explain the importance of using the analogical reasoning sheet for homework planning.
- Mention that the homework project for Logo lab #2 will be similar to the first project, except that it will require variables. (more fully explained in lab)

Administer End of Period Mini-Quiz (if time)

241
Lecture Day 6
Group A
General Outline

Review Two Variable Procedures

On Computer- •Briefly show again the procedure of:
 To Rectangle :W :L
 Repeat 2 [Fd :W Rt 90 Fd :L Rt 90]
 End

Introduce Recursion

On Computer- •Type the following, ask students to predict output:
 (explain that procedure calls itself, etc....)

To Boxes :X (Square :X already in editor)
Square :X
Boxes :X - 10 (Use initial input of 50, etc...)
End

•Add a conditional statement to stop the recursion:

If :X < 0 [Stop] (placed after To Boxes :X line)

•Show the following, ask students to predict output:
(already typed in on demo disk)

To Stack :X To Move :X
Square :X Fd :X
Move :X End
Stack :X - 10
End

•Add stop statement

If :X < 0 [Stop] (placed after To Stack :X line)

Activity Sheet #LA6 (pass out sheet now)

Transparency- •Work through reasoning sheet with students
 (ENCODE, MAP, INFER, APPLY)

On Computer- •Show a student example

 •Emphasize the move statement, and modularity

Handout-

 •Pass out student answer sheet for recursive houses.

Discuss Homework Assignment #3

Transparency- •Show typical Example Homework Assignment #3
 •Review how to correctly use the planning sheet
 •Review the necessity to turn in the completed
 planning sheet

Administer End of Period Mini-Quiz (only if time)

- Allow use of previous activity sheet

APPENDIX H: CONTROL LARGE GROUP INSTRUCTOR OUTLINES

244
Lecture Day 1
Group B
General Outline

Attendance:

- 1) Verbally ask students to make sure that they are in the A group by checking their schedules, or if needed, the master lists at the front of the room.
- 2) Pass around the split lecture attendance sheets.

Announcements:

- 1) Remind students that split labs start Wednesday September 28.
- 2) Remind students to quickly purchase LogoWriter Disks in IRC.
- 3) Other: _____

Introduce Logo and the Logo Philosophy

- Transparency- •Introduce Logo and the Logo philosophy.
 •Indicate that we will discuss the skill of analogical reasoning in more detail as a skill typical of programming.

Introduce Analogical Reasoning

- Transparency- •Define analogical reasoning
Transparency- •Discuss examples of analogical reasoning

Illustrate the Process of Analogical Reasoning

- Transparency- •Mention that analogical is a very global skill, but that it has been attempted to be duplicated when people take tests using analogies. Typically, what is involved in such a test is the process shown.

Relate Analogical Reasoning to Programming

- Transparency- •Discuss how expert programmers use analogical reasoning.

Introduce LogoWriter

- Boot up Disk- •Show initial entry screen, choosing new page
 •Show the following primitives:
(asking students to take notes, and predict outcome to typed primitive command)

FD xx	RT xx	PU	CG	HT
BK xx	LT xx	PD	HOME	ST

Distribute End of Period Mini-Quiz (if time)

Pass out the quiz, allow students to answer right on the half sheet of paper. Remind students to place name, lab section, and A/B group on quiz.

245
Lecture Day 2
Group B - General Outline

Attendance:

- 1) Verbally ask students to make sure that they are in the A group by checking their schedules, or if needed, the master lists at the front of the room. (right before class starts)
- 2) Pass around the split lecture attendance sheets.

Announcements:

- 1) Remind students that split labs start Wednesday September 28.
- 2) Remind students to quickly purchase LogoWriter Disks in IRC.
- 3) Other: _____

Review Primitives

- On Computer-
- Review briefly booting up LogoWriter as it boots.
 - Review briefly some primitives (emphasize PU, PD, CG, HOME)
 - Discuss primitive sequence for drawing a square

Introduce the Repeat Command

- On Computer-
- Introduce the repeat command with a square:
Repeat 4 [Fd 50 Rt 90]
 - Ask students to predict what happens when a change is made:
Repeat 4 -----> Repeat 8

Introduce Procedures

- On Computer-
- Show how to define procedures by use of a square:
(Using open-apple-f editor)

To Square
Repeat 4 [Fd 50 Rt 90]
End

- Change square within the editor to a different size.
Change Fd 50 -----> Fd 30
- Show that the computer now knows a new word by:
Repeat 5 [Square Rt 45]

Activity Sheet L2B - (pass out sheet now)

(Students should always write on these sheets in case of a quiz, which may or may not be open notes!)

- Transparency-
- Ask students to write a procedure for a triangle

- On Computer-
- Show a student example
(if running out of time use the transparency answer)

- Mention or discuss the angle of 120 degrees

Namepage Command and Saving

On Computer-

- Show namepage command of: Np "XXXXXX
- Discuss necessity of hitting escape

247
Lecture Day 3
Group B - General Outline

Attendance:

- 1) Ask students to make sure that they are in the A group by checking their schedules, or if needed, the master lists at the front of the room. (*right before class starts*)
- 2) Pass around the split lecture attendance sheets.

Announcements:

- 1) Remind students that split labs start tomorrow Wednesday Sept. 28.
(*Lists at the front of the room will show where students need to go*)
- 2) Remind students to purchase LogoWriter Disks in the IRC before lab.
- 3) Other: _____

Review Procedures

On Computer- •Briefly show again the following procedures:
(*including getting in and out of the editor*)

To Square	To Triangle
Repeat 4 [Fd 50 Rt 90]	Repeat 3 [Fd 50 Rt 120]
End	End

Introduce Procedures within Procedures

On Computer- •Show procedures can be placed within procedures:

To Stack	(Square already in the editor)
Square	
Fd 50	
Square	
End	

•Show that a staggered stack could be made with:
(*placing positioning commands in a move procedure*)

To Stack (with)	To Move
Square	Fd 50
Move	Rt 90
Square	Fd 25
End	Lt 90
	End

Activity Sheet #LB3 (pass out sheet now)

Transparency- •Ask students to try and write a procedure for a house.

On Computer- •Show a student example and discuss

Transparency- •Debrief the prepared answer shown on the transparency
•Emphasize the move statement, and modularity

Discuss Homework Assignment #1

(all three of the following will be passed out and rediscussed in lab)

- Transparency- •Show typical Example Homework Assignment #1
 •project should be as extensive, run with one
 command, and be broken into parts.
- Transparency- •Show the required student planning sheet.
 •STUDENTS WILL USE THIS SHEET
 INTERACTIVELY TO RECORD THEIR PROJECT
 WHILE DEVELOPING IT ON THE COMPUTER.
- Transparency- •Show the grading criteria sheet for the project.
 •students need to turn in a project on disk,
 planning sheet, and criteria sheet.

Administer End of Period Mini-Quiz

249
Lecture Day 4
Group B- General Outline

Attendance:

- 1) Ask students to make sure that they are in the B group by checking their schedules, or if needed, the master lists at the front of the room. (*right before class starts*)
- 2) Pass around the split lecture attendance sheets.

Announcements:

- 1) Remind students they should be attending split labs now.
(*Lists at the front of the room will show where students need to go*)
- 2) Remind students to purchase LogoWriter Disks in the IRC before lab.
- 3) Remind students that their first LogoWriter project will be due at the start of their second LogoWriter Lab, and that three things will need to be turned in:
 - 1) a project on disk of at least 6 procedures
 - 2) a planning sheet with a written copy of the program
 - 3) a criteria sheet with the name of the project
- 3) Other announcements: _____

Introduce Variables

On Computer-

•Briefly show the square procedure:

To Square
Repeat 4 [Fd 50 Rt 90]
End

discuss that if we want a square of a different size, we need to go in and actually change the procedure or retype with a slightly different name.

To Square :X
Repeat 4 [Fd :X Rt 90]
End

show that this procedure is much more powerful and flexible.

•Show the Boxes procedure using variables:

To Boxes
Square 50 Type: Boxes

Square 40
Square 30 *Thus procedures using variables can be placed within other procedures.*
End

•Modify the Boxes procedure to take input:

To Boxes :X
Square :X Type: Boxes 50, or Boxes 70, etc...

Square :X-10
Square :X-20 *Thus the variable can be passed from an input to the calling procedure to internal subprocedures.*
End

Activity Sheet #LB4 (pass out sheet now)

Handout- •Ask students to write a procedure for drawing a house of any size using variables.

Transparency- •Place a transparency of the past house procedure on the screen in case students would like to look at it.

GIVE NO INITIAL DISCUSSION OF THE TRANSPARENCY, JUST ALLOW STUDENTS TO REFERENCE IT SHOULD THEY DESIRE TO.

On Computer- •Show a typical example (either student's or prepared one)

Transparency- •Debrief the prepared answer shown on the transparency
•Emphasize the syntax format for variable procedures

Mention Homework Assignment #2

- Merely mention that homework assignment number 2 will be similar to the first, but will use variables.
- It will be discussed in more depth next lecture and in lab

Administer End of Period Mini-Quiz (if time)

Review Homework Assignment #2

Transparency- •Show the "stick people" example project using variables, and discuss that the second project will be similar to the first project, except that it will require variables. (more fully explained in lab)

Fill Time (fill extra time with non Logo activity)

Activity- •Perhaps show MultiScribe, or Kings Rule, etc...

Administer End of Period Mini-Quiz (if time)

253
Lecture Day 6
Group B
General Outline

Review Two Variable Procedures

On Computer- •Briefly show again the procedure of:
 To Rectangle :W :L
 Repeat 2 [Fd :W Rt 90 Fd :L Rt 90]
 End

Introduce Recursion

On Computer- •Type the following, ask students to predict output:
 (explain that procedure calls itself, etc.....)

To Boxes :X (Square :X already in editor)
Square :X
Boxes :X - 10 (Use initial input of 50, etc...)
End

•Add a conditional statement to stop the recursion:

If :X < 0 [Stop] (placed after To Boxes :X line)

•Show the following, ask students to predict output:
 (already typed in on demo disk)

To Stack :X To Move :X
Square :X Fd :X
Move :X End
Stack :X - 10
End

•Add stop statement

If :X < 0 [Stop] (placed after To Stack :X line)

Activity Sheet #LB6 (pass out sheet now)

Transparency- •Ask students to write a procedure for
 recursive houses.
 (may leave House program up on computer screen)

On Computer- •Show a student example

 •Emphasize the move statement, and modularity

Handout- •Pass out student answer sheet for recursive houses.

Discuss Homework Assignment #3

Transparency- •Show typical Example Homework Assignment #3

 •Pass out the example sheets for student reference

 •Review necessity to turn in the planning sheet also

Administer End of Period Mini-Quiz (only if time)

- Allow use of previous activity sheet

APPENDIX I: EXPERIMENTAL LAB INSTRUCTOR OUTLINES

256
Lab #1
Group A
General Outline

Roll Call, Record Keeping, Announcements

- Ask each student their name and place a check on the roster as they enter.
(please make sure students are in the right place!)
- Collect Appleworks Assignment
(students hand it in as soon as they come in)
(assume students have saved to disk, let them keep their disk)
- Mention to students that part of the instruction will involve turning on and turning off the monitors, so that everyone is doing the same thing

MONITORS ON

Boot up LogoWriter

- Insure all students have a LogoWriter disk, (loan or trade those who don't).
- Have all students boot up LogoWriter, (and start a new page)

Practice with the Primitives

- On Computers- •Allow students to practice using the primitive commands
(about 5 minutes, primitives are on chalkboard)

FD xx	RT xx	PU	HT	HOME
BK xx	LT xx	PD	ST	CG

Practice with the Repeat Statement

- On Computers- •Allow students to practice using the repeat commands
(exploration for about 5 minutes, trying these examples)
(these should be on the chalkboard also)

Repeat 4 [Fd 50 Rt 90]	(square)
Repeat 3 [Fd 50 Rt 120]	(triangle)
Repeat 2 [Fd 50 Rt 90 Fd 100 Rt 90]	(rectangle)

Review Procedures **MONITORS OFF**

- Demonstration- •Show how to enter the "open-apple-F" editor
•Show again how to build these procedures:

To Square	To Triangle
Repeat 4 [Fd 50 Rt 90]	Repeat 3 [Fd 50 Rt 120]
End	End

MONITORS ON

- On Computers- •Have students enter and test the Square and Triangle
(about 5 min)

MONITORS OFF**Analogical Reasoning Sheet #SA1 (pass out sheet now)**

Transparency- •Work through sheet step by step with student discussion
(ENCODE, INFER, MAP, APPLY) *Students not on computer*

MONITORS ON

On Computers- •Have students try their program on the computer
(students keep a record on the activity sheet)

MONITORS OFF

Discussion- •Show the transparency of a typical answer
•Review the procedure and respond to questions

Review Procedures within Procedures

Demonstration- •Review how to use procedures in procedures with:

To Stack	To Rectangle
Rectangle	Repeat 2 [Fd 25 Rt 90 Fd 50 Rt 90]
Move	End
Rectangle	
End	To Move
	Fd 25
	End

MONITORS STILL OFF**Analogical Reasoning Sheet #SA2 (pass out sheet now)**

Transparency- •Work through sheet step by step with student discussion
(ENCODE, INFER, MAP, APPLY) *Students not on computer*

MONITORS ON

On Computers- •Have students try their program on the computer
(students keep a record on the activity sheet)

MONITORS OFF

Discussion- •Show the transparency of a typical answer
•Review the procedure and respond to questions

MONITORS STILL OFF**Review Homework Assignment #1**

- Insure that students have a homework planning sheet
- Insure that students have a homework grading sheet
(will hand in disk, planning sheet, & grading sheet)
- Discuss what will need to be handed in for a grade

Transparency- •Show Example Homework Assignment #1 Transparency

**STUDENTS MUST SHOW A FAIRLY COMPLETE
PLANNING SHEET TO THE LAB INSTRUCTOR BEFORE
BEING ALLOWED TO TURN ON THEIR MONITOR AND
BEGIN TO WORK ON THE COMPUTER**

MONITORS ON

Allow students to work on Homework

For rest of period, work on respective homework.

To save use NP "lastname1 command, and press escape

259
Lab Day 2
Group A
General Outline

Administrative:

- Hand back any papers that need to be returned to students.
- Have students turn in their LogoWriter projects by:
 - 1) Booting up their project so that it shows on the screen.
 - 2) Making sure that the page holding their project is named with their "lastname1", if not, they need to rename the page with this name.
 - 3) Inserting the instructor's master disk, and pressing escape (this saves it on the instructor's disk)
 - 4) Students must turn in their planning and criteria sheets, but will keep their own LogoWriter disk.

Review Variables

MONITORS OFF

Demonstration- •Briefly show the variable square and triangle procedures:
To Square :X To Triangle :X
Repeat 4 [Fd :X Rt 90] Repeat 3 [Fd :X Rt 120]
End End

•Mention that to run these you must type Square 50, etc...

MONITORS ON

On Computers- *Have students enter and test the square and triangle procedures using variables.

MONITORS OFF

Demonstration- •Discuss the Stack procedure using variables:
(type it in and ask students for a prediction of output when Stack 40 is run)

To Stack :X
Square :X (Square :X already in editor)
Fd :X
Square :X - 10
Fd :X -10
Square :X - 20
End

Activity Sheet #SA3 (pass out sheet now)

Transparency- •Lead students in a discussion of steps on the sheet
•Also ask students which "step" comes next, and what that step entails, before the discussion used for each step.
(ENCODE, INFER, MAP, APPLY)

•Emphasize how looking back at a past problem helps

MONITORS ON

- On Computer-
- Let students test their written program on the computer
 - Students should try to keep a record of the output as on sheet

MONITORS OFF

- Transparency-
- Discuss with students the example answer to this sheet.

Review Procedures with Two Variable Input

- Demonstration-
- Briefly show and discuss the variable rectangle procedure

```
To Rectangle :W :L
Repeat 2 [Fd :W Rt 90 Fd :L Rt 90]
End
```

Activity Sheet #SA4 (pass out sheet now)

- Transparency-
- Lead students in a discussion of steps on the sheet
 - Also ask students which "step" comes next, and what that step entails, before the discussion used for each step. (ENCODE, INFER, MAP, APPLY)

- Emphasize how looking back at a past problem helps

MONITORS ON

- On Computer-
- Let students test their written program on the computer
 - Students should try to keep a record of the output as on sheet

MONITORS OFF

- Transparency-
- Discuss with students the example answer to this sheet.

Review Homework Assignment #2

- Distribute the homework planning sheets and criteria sheets.

- Transparency-
- Discuss what is expected for Homework assignment #2 by going over the planning and criteria sheets.

STUDENTS MUST SHOW THE LAB INSTRUCTOR A GRAPHIC PICTURE AND PLANNED CALLING PROCEDURE BEFORE BEGINNING THEIR PROJECT ON THE

COMPUTER.

Note:

Students will only be required to write out the calling procedure on their planning sheet, no other details will be necessary to turn in! Mention that on the midterm, students will probably not have enough time to write everything out before typing it in, so students should try to continue their careful planning, but without the necessity of writing everything out in pencil first.

Allow students to work on Homework

- For rest of period, work on respective homework
- Students will need to turn in a project on disk, a planning sheet with graphic and calling procedure, and a criteria sheet.

Make sure students save at least once while in lab.

- Students should leave lab with at least part of their project saved under a page named with "lastname2"

262
Lab Day 3
Group A
General Outline

Review Recursion

Demonstration- •Briefly show the recursive procedure:
(ask for a prediction of what it does)

To Boxes :X
Square :X
Boxes :X-10
End

To Square :X
Repeat 4 [Fd :X Rt 90]
End

*Ask students what is occurring.
(Boxes is calling itself, etc...)

*Add a stop statement: If :X<10 [Stop]

Demonstration- •Now show the following recursion example:
(Ask students for output predictions, given specific input)

To Coil :X
If :X < 1 [Stop]
Circle
Move
Coil :X-1
End

To Circle
Repeat 36 [Fd 2 Rt 10]
End

To Move
Pu
Fd 20
Pd
End

(use Coil 5, Coil 4, etc...)

On Computers- •Have students try to type in and run the coil procedure.

Activity Sheet #SA5 (pass out sheet now)

Transparency- •Lead students in a discussion of steps on the sheet
(ENCODE, INFER, MAP, APPLY) *Students not on computer*
•Emphasize how looking back at a past problem helps

On Computer- •Let students test their written program on the computer
•Students should keep a record of the output as on sheet

Transparency- •Discuss with students the example answer to this sheet.

A Brief Discussion of Analogical Reasoning

Discussion- •Ask students the following Questions

- 1) How do these these sheets compare to what a good programmer does?
- 2) Again, what is "analogical reasoning"?

- 3) What sorts of occupations might depend on analogical reasoning? (Doctors, Carpenters, etc.....)

Review Homework Assignment #3

Transparency-

- Show typical Example Homework Assignment #3
- EMPHASIZE how to use planning sheet '
(Handout given in lecture, extras available)
- Must hand in program and planning sheet for full credit
- To save use NP "lastname3 command, and press escape

Allow students to plan Homework

- Students must show Lab Instructor a fairly completed planning sheet before beginning to work on the computer.

Allow students to work on Homework

- For rest of period, work on respective homework.

· APPENDIX J: CONTROL LAB INSTRUCTOR OUTLINES

265
Lab #1
Group B
General Outline

Roll Call, Record Keeping, Announcements

- Ask each student their name and place a check on the roster as they enter.
(please make sure students are in the right place!)
- Collect Appleworks Assignment
(students hand it in as soon as they come in)
(assume students have saved to disk, let them keep their disk)
- Mention to students that part of the instruction will involve turning on and turning off the monitors, so that everyone is doing the same thing

MONITORS ON

Boot up LogoWriter

- Insure all students have a LogoWriter disk, (loan or trade those who don't).
- Have all students boot up LogoWriter, (and start a new page)

Practice with the Primitives

- On Computers- •Allow students to practice using the primitive commands
(about 5 minutes, primitives are on chalkboard)

FD xx	RT xx	PU	HT	HOME
BK xx	LT xx	PD	ST	CG

Practice with the Repeat Statement

- On Computers- •Allow students to practice using the repeat commands
(exploration for about 5 minutes, trying these examples)
(these should be on the chalkboard also)

Repeat 4 [Fd 50 Rt 90]	(square)
Repeat 3 [Fd 50 Rt 120]	(triangle)
Repeat 2 [Fd 50 Rt 90 Fd 100 Rt 90]	(rectangle)

Review Procedures

MONITORS OFF

- Demonstration- •Show how to enter the "open-apple-F" editor
•Show again how to build these procedures:

To Square	To Triangle
Repeat 4 [Fd 50 Rt 90]	Repeat 3 [Fd 50 Rt 120]
End	End

MONITORS ON

- On Computers- •Have students enter and test the Square and Triangle
(about 5 min)

MONITORS STILL ON**Class Activity Sheet #SB1 (pass out sheet now)**

- On Computers- •Have students try to develop a program for the shape of the figure on the activity sheet. They should be encouraged to work immediately on the computer.
- Encourage students to keep a record of output attempts and any other notes on the activity sheet.

MONITORS OFF

- Discussion- •Show the transparency of a typical answer
•Review the procedure and respond to questions

Review Procedures within Procedures

- Demonstration- •Review how to use procedures in procedures with:

To Stack	To Rectangle
Rectangle	Repeat 2 [Fd 25 Rt 90 Fd 50 Rt 90]
Move	End
Rectangle	
End	To Move
	Fd 25
	End

MONITORS ON**Activity Sheet #SB2 (pass out sheet now)**

- On Computers- •Have students try to develop a program for the shape of the figure on the activity sheet. They should be encouraged to work immediately on the computer.
- Encourage students to keep a record of output attempts and any other notes on the activity sheet.

MONITORS OFF

- Discussion- •Show the transparency of a typical answer
•Review the procedure and respond to questions

MONITORS STILL OFF**Review Homework Assignment #1**

- Insure that students have a homework planning sheet
•Insure that students have a homework grading sheet
(will hand in disk, planning sheet, & grading sheet)
•Discuss what will need to be turned in for a grade
- Transparency- •Show Example Homework Assignment #1 Transparency

MONITORS ON

- Encourage students to begin work on the computer immediately. They can write out their planning sheet at any time, it just must eventually be completed as part of the overall assignment.
- For rest of period, work on respective homework.
- To save use NP "lastname1 command, and press escape

268
Lab Day 2
Group B
General Outline

Review Variables

Demonstration- •Briefly show the variable square procedure:
To Square :X
Repeat 4 [Fd :X Rt 90]
End

On Computers- *Have students enter and test a square procedure which uses variables.

Demonstration- •Discuss the Stack procedure using variables:
(type it in and ask students for a prediction of output)

```
To Stack :X
Square :X      (Square :X already in editor)
Fd :X
Square :X - 10
Fd :X - 10
Square :X - 20
End
```

On Computers- •Have students try to type in and run the stack procedure.

Activity Sheet #SB3 (pass out sheet now)

On Computers- •Have students try to develop a program for the shape of the figure on the activity sheet. They may either work directly on the computer or write it in pencil first.

•Encourage students to keep a record of output attempts (bottom part of the sheet)

Discussion- •Discuss with students, the example answer to this sheet

Review Procedures with Two Variable Input

Demonstration- •Briefly show and discuss the variable rectangle procedure

```
To Rectangle :W :L
Repeat 2 [Fd :W Rt 90 Fd :L Rt 90]
End
```

•Have students type in and try this procedure

Activity Sheet #SB4 (pass out sheet now)

On Computers- •Have students try to develop a program for the shape of the figure on the activity sheet. They may either work directly on the computer or write it in pencil first.

•Encourage students to keep a record of output attempts (bottom part of the sheet)

Discussion- •Discuss with students, the example answer to this sheet

Review Homework Assignment #2

Transparency- •Show typical Example Homework Assignment #2
•Mention that students may want to plan first with sheet
(*Handout given in lecture, extras available*)

•MUST hand in program and planning sheet for full credit!

•To save use NP "lastname2 command, and press escape

Allow students to work on their Homework

•Students may either work immediately on the computer, or use the planning sheet first. However, the planning sheet must be completed when it is handed in.

•Rest of period, students work on respective homework.

270
Lab Day 3
Group B
General Outline

Review Recursion

Demonstration- •Briefly show the recursive procedure:
(ask for a prediction of what it does)

To Boxes :X
Square :X
Boxes :X-10
End

To Square :X
Repeat 4 [Fd :X Rt 90]
End

*Ask students what is occurring.
(Boxes is calling itself, etc...)

*Add a stop statement: If :X<10 [Stop]

Demonstration- •Now show the following recursion example:
(Ask students for output predictions, given specific input)

To Coil :X
If :X < 1 [Stop]
Circle
Move
Coil :X-1
End

To Circle
Repeat 36 [Fd 2 Rt 10]
End

To Move
Pu
Fd 20
Pd
End

(use Coil 5, Coil 4, etc....)

On Computers- •Have students try to type in and run the coil procedure.

Activity Sheet #SB5 (pass out sheet now)

On Computers- •Have students try to develop a program for the shape of
the figure on the activity sheet. They may either work
directly on the computer or write it in pencil first.

•Encourage students to keep a record of output attempts
(bottom part of the sheet)

Discussion- •Discuss with students, the example answer to this sheet

A Brief Discussion of Logo

Discussion- •Ask students the following Questions

- 1) How could you use LogoWriter in the classroom?
(contests, projects, etc.....)
- 2) Does using LogoWriter in pairs help or hurt students?
- 3) How old should students be using LogoWriter?
(actually any age, etc.....)

Review Homework Assignment #3

Transparency- •Show typical Example Homework Assignment #3
•Mention that students may want to plan first with sheet
(*Handout given in lecture, extras available*)

•MUST hand in program and planning sheet for full credit!
•To save use NP "lastname3 command, and press escape

Allow students to work on their Homework

- Students may either work immediately on the computer, or use the planning sheet first. However, the planning sheet must be completed when it is handed in.

- Rest of period, students work on respective homework.

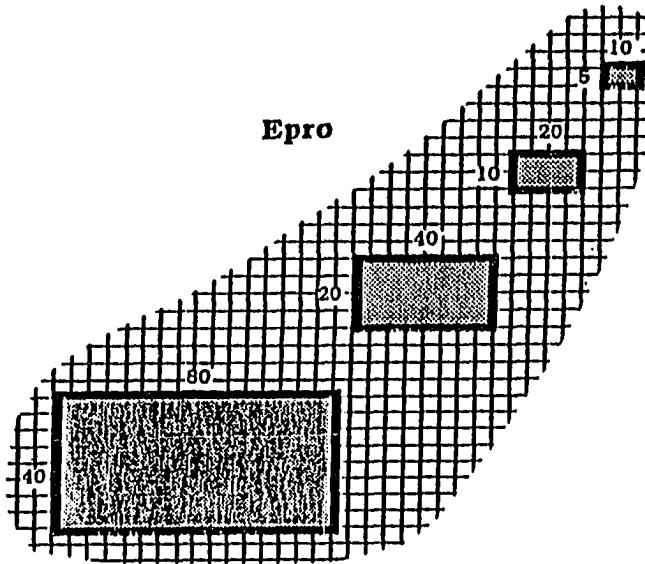
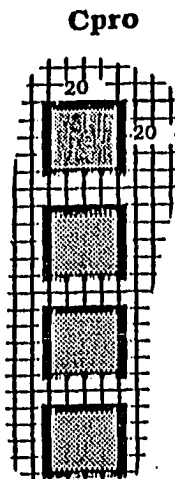
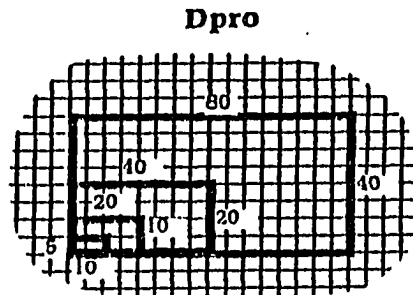
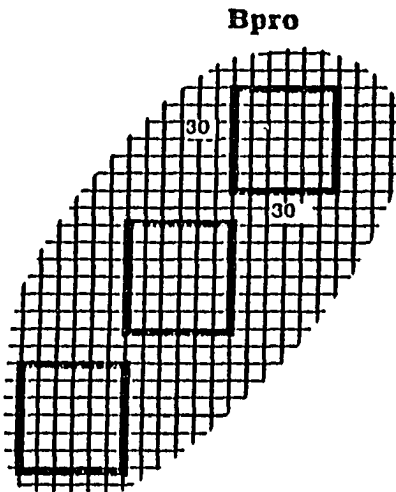
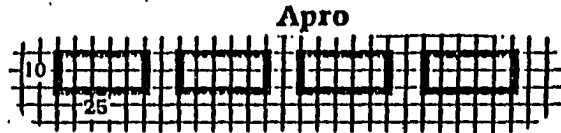
APPENDIX K: REUSE OF SUBPROCEDURES PROGRAMMING TEST

LogoWriter Production Test for SecEd 101

Directions:

- 1) All five problems must be done on the same single page in LogoWriter.
- 2) The page will be named by use of NP "lastname.M
- 3) Graphics can be anywhere on the screen but can not wrap around the screen.
- 4) Problems must be done in order and each main procedure for each problem should be called To Apro, To Bpro, To Cpro, To Dpro, To Epro, etc...
- 5) Remember, procedures should be well written and modular in structure.
- 6) When you wish to save, as always, merely press escape; you will hand in the disk that your midterm is saved on.

One grid square, \square , = 5 turtle steps on each side



APPENDIX L: LOGOWRITER BASIC COMPREHENSION TEST

Introductory LogoWriter Basic Comprehension Test
(Turtle Graphics)

The following is a list of general objectives tested by this test. The test is designed to examine the basic knowledge and understanding of some fundamental Logo commands and concepts. This test is targeted at the Bloom Taxonomy levels of Knowledge and Comprehension only, and does not attempt to measure higher levels of learning. Higher order programming concepts such as modularity, and top-down design, are utilized in the test questions, but are not targeted specifically for evaluation.

Basic Objectives:

1. Basic Turtle Commands (Primitives)

- 1.1) The student is able to identify the function of primitive commands.
- 1.2) The student is able to differentiate between pre-defined primitive commands, and user defined procedures, within the Logo language.
- 1.3) The student is able to predict changes in the turtle's state, (heading and position), implemented by sequences of primitive commands.
- 1.4) The student is able to predict the graphical output produced by sequences of primitive commands.

2. Repeat Commands

- 2.1) The student is able to identify the proper syntax of the repeat command.
- 2.2) The student is able to select an equivalent repeat statement for a repeated sequence of primitive commands.
- 2.3) The student is able to recognize that the repeat statement is a more efficient and simplified structure for repeated sequences of primitives or procedures.
- 2.4) The student is able to predict the output effect of the repeat command used with primitives and defined procedures.

3. Basic Procedures

- 3.1) The student is able to identify the proper syntax for defining a procedure.
- 3.2) The student is able to recognize that a procedure is basically a set of command steps defined to perform some task.
- 3.3) The student is able to predict the output effects of procedures using sequenced primitive commands and the repeat command.
- 3.4) The student is able to predict the output effects of procedures when used in combination with primitive and repeat commands.
- 3.5) The student is able to identify operational features of the LogoWriter Editor.

4. Super-Procedures and Sub-Procedures

- 4.1) The student is able to differentiate between the main calling procedure and its subprocedures in a program.
- 4.2) The student is able to identify that the restructuring of a larger procedure into a calling procedure and subprocedures promotes effective programming by problem analysis, task division, and procedure reusability.
- 4.3) The student is able to predict the graphic effects of the execution of a calling procedure with its included subprocedures.
- 4.4) The student is able to select a clear, concise, calling procedure that calls appropriate sub-procedures.

5. Variable Use

- 5.1) The student is able to recognize the proper syntax for procedures using single variable and dual variable inputs.
- 5.2) The student is able to recognize that variables are placeholders for changeable values that permit flexibility and generality in procedures.
- 5.3) The student is able to predict the graphic effects of the execution of procedures using variables with specific input values.
- 5.4) The student is able to predict the graphic effects of the execution of procedures using variables, with internal modification of variables, given specific input to the procedures.
- 5.5) The student is able to select an appropriate procedure for a programming problem requiring the use of more than one variable.

6. Recursive Procedures and Conditional Statements

- 6.1) The student is able to identify the proper syntax and format of a procedure using recursion.
- 6.2) The student is able to recognize that a recursive procedure is a procedure which calls itself as a subprocedure permitting modifiable repetition.
- 6.3) The student will be able to predict the graphic effects of the execution of basic procedures using recursion.
- 6.4) The student will be able to predict the graphic effects of the execution of procedures using recursion and conditional statements.
- 6.5) The student is able to select an appropriate stop procedure for a recursion.

Secondary Education 101
Midterm Test - LogoWriter Comprehension Part

Name _____

Directions: Please read the following questions carefully and select the best answer for each question. In questions involving graphics, or sequences of specific commands, always assume that the turtle starts in the home position unless the question states otherwise.

1. Examine the following primitive command descriptions; which of the descriptions are incorrect?

Fd - moves the turtle forward a certain distance
Rt - turns the turtle to the right a certain number of degrees
Home - clears the screen and moves the turtle to the screens center facing up.
Fill - fills a graphic shape with a specific color
Pu - picks up the drawing pen of the turtle so that no line is drawn as the turtle moves


- a. all of the descriptions are correct.
- b. one of the descriptions is incorrect.
- c. two of the descriptions are incorrect.
- d. three descriptions are incorrect.
- e. the descriptions are all basically correct, but the primitive commands must be typed in all capital letters for them to work.

2. In the LOGO programming Language, which of the following is not a primitive?





- a. Cg
- b. Fd
- c. Seth
- d. Fillit
- e. Home

3. In Logo, the "primitive" commands are:

- a. Useful procedures invented and defined by the user to perform some task, like moving the turtle forward or drawing a triangle.
- b. Useful procedures that are already defined in the Logo language when it starts up.
- c. The basic movement commands of FD, BK, RT, and LT, which are the only commands that actually move the turtle on the screen, and thus the only "primitive" commands.
- d. The commands of PU, PD, PE, Home, HT, ST, and CO, which are the only commands that require no input numbers, thus they are the only "primitive" commands.
- e. None of the above statements is correct.

4. Given the following sequence of primitive commands, and the information that the turtle is facing directly to the right of the screen, () before the commands are executed, which way does the turtle face after the commands are executed?

```
Fd 50
Rt 90
Fd 100
Rt 180
Bk 40
Lt 90
```

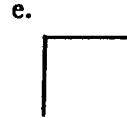
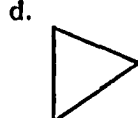
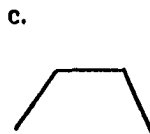
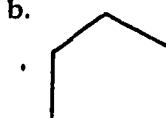
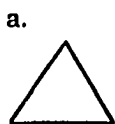
- a. The turtle now faces to the bottom of the screen. 
 b. The turtle now faces to the left of the screen. 
 c. The turtle now faces to the top of the screen. 
 d. The turtle still faces to the right of the screen. 
 e. It is impossible to tell without specific coordinates.

5. Which of the following sets of commands will position the turtle the greatest distance away from the home position?
 (assume that the turtle starts in the home position)

- | | | | | |
|---|--|--|---|--|
| a. Fd 100
Bk 100
Rt 90
Fd 100
Bk 40 | b. Fd 200
HT
Fd 100
Home
Fd 20 | c. Bk 100
Rt 90
Ht
Rt 90
Fd 70 | d. Fd 100
BK 200
Fd 25
Ht
Fd 50 | e. It is impossible to tell without typing these commands into the computer. |
|---|--|--|---|--|

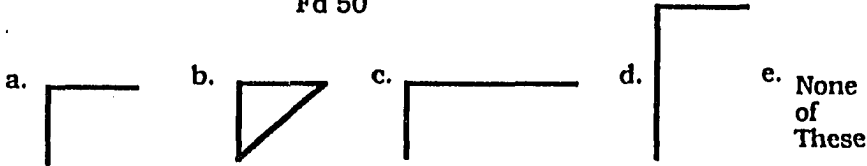
6. What will the following sequence of commands draw?
 (assume that the turtle starts in the home position)

```
Fd 50
RT 60
FD 50
RT 60
Fd 50
RT 60
```



7. What will be drawn by the following sequence of commands?
(assume that the turtle starts in the home position)

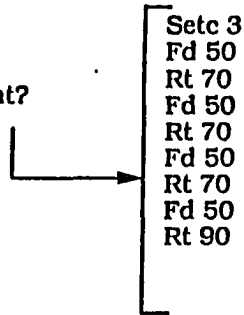
Fd 50
Rt 90
Fd 50
Home
Fd 50



8. Which of the following Repeat commands will not produce an error message when it is executed?

- a. Repeat [Fd 50 Bk 50 Rt 60]
- b. Repeat Fd 50 [Rt 90]
- c. Repeat 3 (Fd 50 Bk 50)
- d. Repeat 4 [Pu Rt 90 Fd 50 Pd Bk 50]
- e. All of the above statements will produce error messages.

9. Which of the choices below is the most efficient replacement for this set of commands to the right?



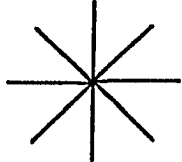
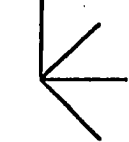

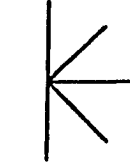
- a. Setc 3
Repeat 3 [Fd 50 Rt 70]
Fd 50
Rt 90
- b. Setc 3
Repeat 4 [Fd 50 Rt 70]
Rt 90
- c. Setc 3
Fd 200
Rt 210
Rt 90
- d. Repeat 3 [Setc 3 Fd 50 Rt 70]
Fd 50
Rt 90
- e. Repeat 3 [Setc 3 Fd 50 Rt 70]

10. In Logo, the Repeat command:

- will make the turtle do something exactly twice, (for instance: Repeat Square draws two squares exactly the same).
- provides the capability to simplify repeated sequences of commands into a single more efficient command.
- must be used when drawing a square, triangle, rectangle, or circle.
- will make the turtle do something over and over forever, until the programmer presses the "open-apple" and "S" keys.
- none of the above are correct.

11. What shape would the following repeat command draw? (assume that the turtle starts in the home position) ▲

Repeat 5 [Fd 50 Bk 50 Rt 45]

- a. 
- b. 
- c. 
- d. 
- e. None of These

12. Which of the following procedures will not produce an error message when the procedure is executed?

- | | | | | |
|---|---|---|---|---|
| a. Vee
Lt 45
Bk 50
Rt 90
Fd 50
End | b. To
Vee
Lt 45
Bk 50
Rt 90
Fd 50
End | c. To Vee
Lt 45
Bk 50
Rt 90
Fd 50
Stop | d. To Vee
Lt 45
Bk 50
Rt 90
Fd 50
To End | e. all of these
will produce
error messages |
|---|---|---|---|---|

13. In LogoWriter, the term "Procedure" basically stands for:

- the technique for drawing step by step pictures with a computer
- a set of defined command steps to perform some task
- the important problem solving steps of defining the problem, choosing a plan, carrying out the plan, and looking back at the solution.
- all the important commands for using the editor, such as "open-apple-f"
- none of the above

14. Which of the following procedures would correctly draw the figure shown below? (assume that the turtle starts in the home position)

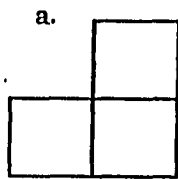
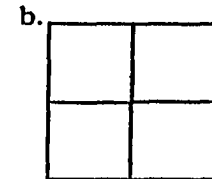
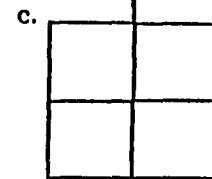
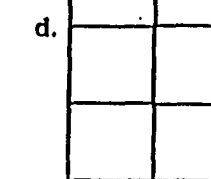


- a. To Peak
RT 45
Fd 50
RT 45
Fd 50
End
- b. To Peak
Fd 50
RT 90
Fd 50
End
- c. To Peak
RT 45
Fd 50
RT 90
Fd 50
End
- d. To Peak
Rt 90
Fd 50
Rt 45
Fd 50
End
- e. None of These

15. Given the Square procedure, what would be the graphical result of the following sequence of commands? (assume that the turtle starts in the home position)

Command Sequence:
Cg
Repeat 4 [Square Rt 90]
Fd 50
Square

To Square
Repeat 4 [Fd 50 Rt 90]
End

- a. 
- b. 
- c. 
- d. 
- e. None of These

16. When using the LogoWriter editor, it is important to:

- a. press "open-apple-f" when entering the editor and "escape" when exiting the editor.
- b. begin every student defined procedure with the word "To" and end every student defined procedure with the word "End".
- c. begin a brand new page for each new procedure.
- d. none of the above are correct.
- e. all of the above are correct.

17. The following is an example of a program in LogoWriter:

```
To Blossom      To Stem   To Flower  To Square
Repeat 10 [Square Rt 36] Home   Stem      Repeat 4 [Fd 50 Rt 90]
End            End       Blossom    End
                End       End
```

Which of the following statements is true?

- Blossom is the main calling procedure for this program.
- Square is the main calling procedure for this program.
- Flower is the main calling procedure for this program.
- Stem and Blossom are both main calling procedures for this program.
- There is no main calling procedure for this program.

18. What is one of the reasons that a programmer might want to divide up a procedure into a calling procedure and various sub-procedures?


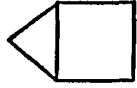


- Because the LogoWriter editor only works with small procedures of no more than one screen long.
- Because it is easier to analyze a problem, and program its solution, in parts.
- Because sub-procedures like Square, Triangle, and Circle are already built into the Logo language, and these won't have to be created by the programmer.
- Because in Logo there is no immediate mode, and the turtle can not execute a command unless it is written into a sub-procedure stored in the editor.
- None of the above are true.

19. Given the following procedures in the workspace, what would be the graphic output when running the procedure "House"? (assume that the turtle starts from the home position)

```
To House
Square
Roof
End
```

```
To Roof
Repeat 3[Fd 50 Rt 120]
End
```

```
To Square
Repeat 4[Fd 50 Rt 90]
End
```

- a.  b.  c.  d.  e. None of these

20. Using the procedures of Frame, Wheel, & Handlebars, and assuming that each of these procedures draw only a specific shape, what is the super-procedure most likely needed for drawing a bicycle?

- | | | | | |
|---|---|---|--|------------------------|
| a. To Bicycle
Frame
Move1
Repeat 2 [Wheel]
Move2
Handlebars
End | b. To Bicycle
Frame
Wheel
Wheel
Handlebars
End | c. To Bicycle
Frame
Move
Wheel
Move
Wheel
Move
Handlebars
End | d. To Bicycle
Frame
Move1
Wheel
Move2
Wheel
Move3
Handlebars
End | e. None
of
These |
|---|---|---|--|------------------------|

21. Looking at the following procedures, which of the statements below would be considered true?

To Mystery :X Fd :X Rt :X + 90 Repeat 100 [Fd :X Rt :X] End	To Something :X :Y Fd :X RT :Y Repeat 100 [Fd :X Rt :Y] End
---	---

- Both the Mystery procedure and the Something procedure use two variables.
- The :X in the line "To Mystery :X", is unnecessary for input and could be removed.
- The Mystery procedure could be executed by typing Mystery 47.
- The Something procedure could be executed by typing Something 17.
- More than one of these statements is true.

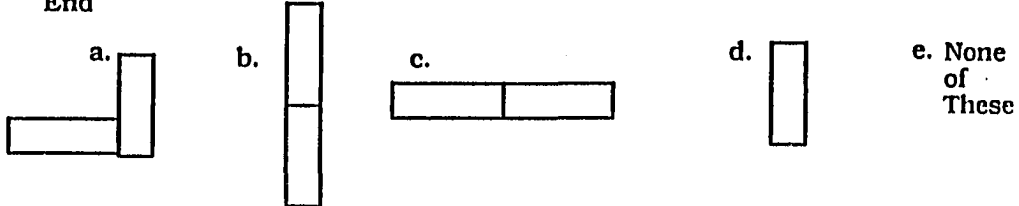
22. One of the reasons programmers may want to use variables in their procedures is because:

- variables are needed in procedures to use the LogoWriter editor .
- variable procedures are what make the graphics in LogoWriter colorful.
- variables are needed for graphics, especially in drawing curved lines.
- procedures using variables are more easily reused in other applications.
- none of the above

23. Using the following procedures, predict what happens when Train 20 50 is executed. (assume the turtle starts in the home position)

```
To Train :Width :Length
  Rectangle :Width :Length
  RT 90
  FD :Length
  LT 90
  Rectangle :Width :Length
End
```

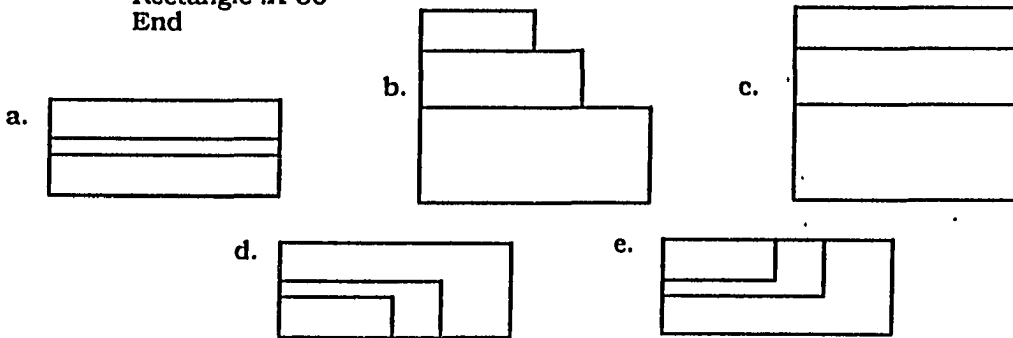
```
To Rectangle :Width :Length
  Repeat 2 [Fd :Width RT 90 FD :Length RT 90]
End
```



24. Which of the figures shown below will result from the execution of Stack 50?

```
To Stack :X
  Rectangle :X
  Rectangle :X-20
  Rectangle :X-30
End
```

```
To Rectangle :X
  Repeat 2 [Fd :X Rt 90 Fd :X * 2 Rt 90]
End
```



25. A student would like to design a LogoWriter program which will draw a triangle placed directly above a square, as in the picture on the right. She would like to have the side of the square and the side of the triangle to be different inputs. Which procedure below, would best fit her desire? (Square and Triangle are already in the workspace)



- a. To Fig :X :Y
Square :X
Fd :X
Triangle :Y
End
- b. To Fig :X
Square :X
Fd :X
Triangle :X
End
- c. To Fig :X :Y
Square :X
Fd :Y
Triangle :Y
End
- d. To Fig :X :Y
Square :X
Triangle :Y
End
- e. None of these would be appropriate

26. Which of the following is an example of a procedure using recursion and a conditional statement to terminate it?

- | | | | | |
|--|---|---|--|------------------------|
| a. To Thing :L
Fd :L
RT 5
Thing :L - 1
IF :L < 0 [Stop]
End | b. To Thing :L
Repeat 4 [Fd :L Rt 5]
Fd :L
If :L < 0 [Stop]
End | c. To Thing :L
For :L = 1 to 4
Fd :L
Rt 90
Next :L
End | d. To Thing :L
Fd :L
Thing :L - 1
End | e. None
of
these |
|--|---|---|--|------------------------|

27. A "recursive" procedure in LogoWriter is a procedure that:

- uses repeated curves within the graphical output.
- is basically the same as a repeat statement but uses less commands.
- calls itself as a sub-procedure.
- calls more than two different sub-procedures.
- all of the above are correct.

28. Looking at the following procedure, which of the statements listed below best describes the execution of the program?

```
To Lots
Repeat 2 [Fd 20 Rt 90 Fd 50 Rt 90]
Pu
Fd 20
Pd
Lots
End
```

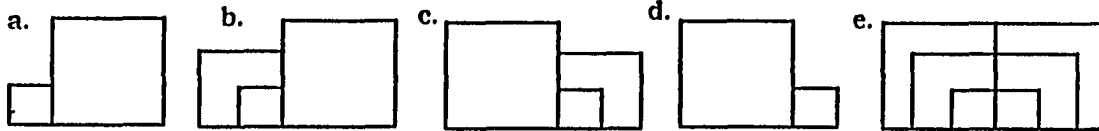
- The procedure draws the same rectangle, in the same place, continually, until someone stops the program.
- The procedure draws two rectangles, one above the other one.
- The procedure draws one rectangle, moves forward, and then gives an error message.
- The procedure continues to draw rectangles stacked above each other until the memory of the computer is filled up.
- None of the statements above describe the execution.

29. Given the procedures shown below, what figure would be drawn by :Mystery 30? (assume that the turtle starts in the home position)

```
To Mystery :S
IF :S = 0 [Stop]
IF :S = 30 [RSquare :S]
IF :S < 20 [LSquare :S]
Mystery :S - 10
End
```

```
To RSquare :S
Repeat 4[FD :S RT 90]
End
```

```
To LSquare :S
Repeat 4[FD :S LT 90]
End
```



30. In the following recursive procedure Blocks, what is the correct conditional statement to stop the procedure so that the output looks like the figure below when Blocks 3 is executed?

Blocks Recursive Procedure

```
(line 1) To Blocks :x
(line 2) Repeat 4 [Fd 50 Rt 90]
(line 3) Fd 50
(line 4) Blocks :x-1
(line 5) End
```

Desired Output

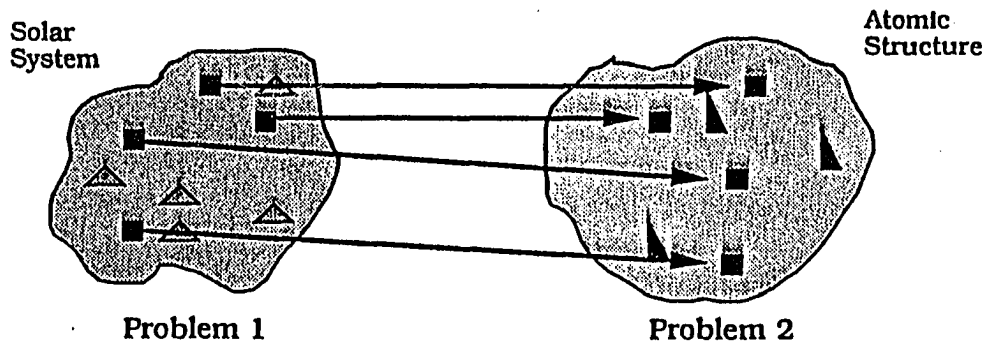


- a. Place the statement: "If :x < 0 [stop]" between lines 1 and 2.
- b. Place the statement: "If :x = 0 [stop]" between lines 1 and 2.
- c. Place the statement: "If :x = 0 [stop]" between lines 3 and 4.
- d. Place the statement: "If :x = 0 [stop]" between lines 4 and 5.
- e. None of the above

**APPENDIX M: ANALOGICAL REASONING INTRODUCTION
TRANSPARENCIES**

Analogical Reasoning:

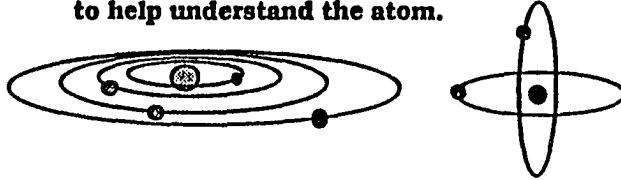
The ability to utilize a well understood problem to provide insight and structure for the development of a solution for a less understood problem.



(Mapping relevant features while ignoring irrelevant features)

Examples of Analogical Reasoning

Using the knowledge of the solar system to help understand the atom.



Referring back to other cars driven in the past to understand how to open the hood of the car you are currently driving.

Past Cars

Inside Hood Release Latch Under Hood Lip
 Double Latch Near Bumper Key Locked

Current Car

Hood Release?

Using your knowledge of Vietnam, to make judgements about the situation in Nicaragua.

US Forces Agression
 Asia Communism
 North/South Vietnamese

US Troop Commitment
 Central America
 Honduras/Nicaragua
 Drug Trafficing

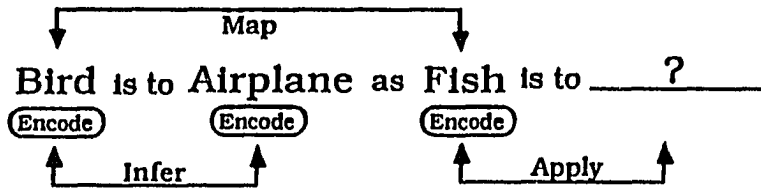
Understanding relationships organized in the form of analogies on intelligence tests.

PISTOL is to BOW as BULLET is to (arrow).



Sternberg Componential Model of Analogical Reasoning

Four basic components to the process.....



- 1. Encoding: Identify characteristics or attributes of each term.**

Bird	Airplane	Fish
Flys	Flys	Swims
Alive	Metal	Alive
Wings	Wings	Gills
Feet	Carries Humans	Resides in Water
etc...	etc...	etc...

- 2. Infering: Relationship looked at between first two terms.**

Birds and **Airplanes** both fly, and have wings to support them in the air, etc...

- 3. Mapping: Relationship looked at between first and third terms.**

Birds and **Fish** both are alive, and travel through environment, etc...

- 4. Applying: Completion of analogy where last term is discovered.**

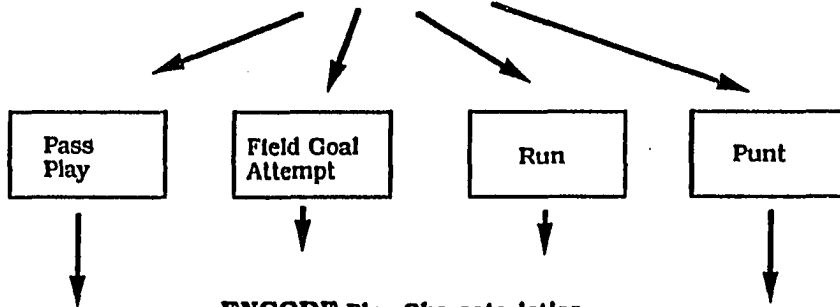
A good answer might be **Submarine**

- moves through water like a fish
- carries humans like an airplane

Coach's Problem:

Fourth down, team behind.
What play to call?

Think of plays used in similar situations:



ENCODE Play Characteristics

<p><u>Pass Play</u> 2 or 3 receivers Flag or Post pattern Risky Long gain possible</p>	<p><u>Field Goal</u> only 3 points high probability close fairly safe usual kicker hurt</p>	<p><u>Run</u> fullback or halfback low gain but steady young fullback strong blocking left</p>	<p><u>Punt</u> loss of football protective measure return possibility good punter</p>
--	---	--	---

INFER Typical Play Outcomes (In past)

<p><u>Pass Play</u> Fairly successful when team not suspecting play</p>	<p><u>Field Goal</u> Has been fairly sure points when up close and good blocking</p>	<p><u>Run</u> Up the middle seems to work best except if team tired</p>	<p><u>Punt</u> Punting effective if time left to resume offense</p>
---	--	---	---

MAP to Current Situation

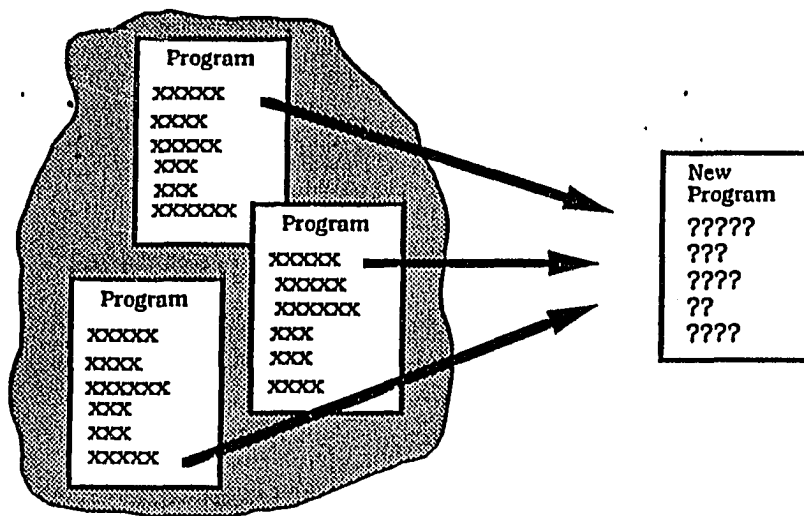
<p><u>Pass Play</u> This team is weak against the pass</p>	<p><u>Field Goal</u> This team has been very successful at blocking field goals</p>	<p><u>Run</u> This team is strong up the middle but we have made yardage on the outside</p>	<p><u>Punt</u> This team never blocks a punt but has excellent returns</p>
--	---	---	--

APPLY Part or All of Old Play as a New Solution.

Make a decision, record result for future use, and if
unsuccessful, no sweat, you'll get them next play. (or year)

Analogical Reasoning in Programming

"an explicit design strategy of expert programmers is the search for similar previously solved problems"

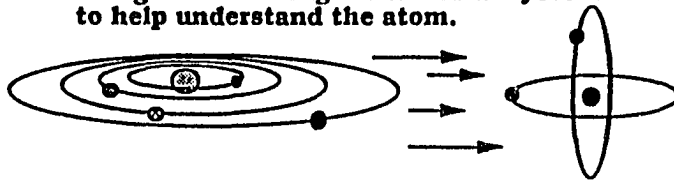


Analogical Reasoning:

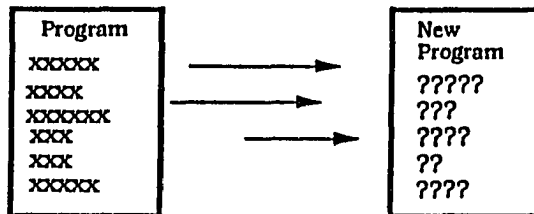
The ability to utilize a well understood problem to provide insight and structure for the development of a solution for a less understood problem.

ENCODE → INFER → MAP → APPLY

Using the knowledge of the solar system to help understand the atom.



Using one computer program to help understand and construct another computer program.



Introduction: Logo & the Logo Philosophy

- Invented by Seymour Papert at MIT
- Educational Programming Environment
 - * Student commands computer (not vice versa)
 - * Exploration and Feedback
 - * Development of powerful ideas & thinking skills

