Instructional Methods for Novice Programmers

By

ANTHONY CLARKE

DIP. Phys. Ed., The University of Melbourne, 1975

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF
THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF ARTS

in

THE FACULTY OF GRADUATE STUDIES

(Department of Mathematics and Science Education)

We accept this thesis as conforming to the required standard

__________________________________________

Dr. Stuart Donn

__________________________________________

Dr. Peter Grimmett

__________________________________________

Dr. Marv Westrom

THE UNIVERSITY OF BRITISH COLUMBIA

© April 1989
In presenting this thesis in partial fulfilment of the requirements for an advanced degree at the University of British Columbia, I agree that the Library shall make it freely available for reference and study. I further agree that permission for extensive copying of this thesis for scholarly purposes may be granted by the head of my department or by his or her representatives. It is understood that copying or publication of this thesis for financial gain shall not be allowed without my written permission.

Department of **EDUCATION (MATH/SCI)**

The University of British Columbia
Vancouver, Canada

Date 25/4/89

DE-6 (2/88)
Abstract

Computer programming is a relatively new phenomenon. Instructional methods in response to this new pedagogy have been many and varied. In the critical interplay between teacher and learner, the learners' perception of the instructional environment is perhaps one of the most important yet least understood variables. Little research has been devoted to understanding the learners' perceptions of the different instructional methods advocated. This study provides an insight into the students' perceptions of two instructional methods, and thus extends the knowledge base for decision making about learning environments for novice programmers.

An introductory computer programming class of sixteen university students was divided into two groups, one to experience the Lecture-lab approach (teacher centred) and the other the Guided Self-discovery approach (student centred). To ensure an even balance of abilities between the two groups, student allocation was based on a test of programming ability and a test of field independence. During the study students completed two attitude questionnaires, three tests of programming ability and a log sheet for every class. At the conclusion of the experimentation period, eight of the sixteen students were interviewed.

The results of the statistical (Pearson's correlation coefficient, t-tests) and interview analysis indicated that novice programmers prefer an instructional framework based on: a teaching style that allows active student participation and substantial student-teacher interactions, a learning environment that allows student exploration and substantial student-student interactions, and resource materials that include regular work sheet.
# Table of Contents

Abstract ........................................... ii

List of Tables .................................... vi

List of Figures .................................... vii

Acknowledgements ................................ viii

Chapter 1 Introduction ............................ 1

1.1 Purpose of the Study .......................... 1
1.2 Need for the Study ............................ 2
1.3 Overview of the Study ....................... 2
   1.3.1 Hypotheses ............................. 3
   1.3.2 Research Variables .................... 4
   1.3.3 Operational Definitions ............... 4
1.4 Chapter Preview .............................. 5

Chapter 2 Review of the Literature .......... 6

2.1 A Relatively Undeveloped Pedagogy ......... 6
2.2 Instructional Approaches ................... 7
   2.2.1 Group Interaction Approach ........... 8
   2.2.2 Discovery Learning or Hands-On Approach 8
   2.2.3 Grammatical Approach ................ 10
   2.2.4 Whole Program or Language Orientation Approach 11
   2.2.5 Bootstrap Learning .................... 12
   2.2.6 Spiral Approach ....................... 13
   2.2.7 Lecture Approach ..................... 14
2.3 The Need for a Suitable Analogy or Model .. 15
2.4 Conclusion .................................. 19
Chapter 3 Methodology

3.1 The Course
3.2 The Instructors
3.3 Measuring Instruments
3.4 Research Design
3.5 Instructional Methods
3.6 Data Collection Procedures
3.7 Data Analysis
    3.7.1 Interview Analysis
    3.7.2 Statistical Analysis

Chapter 4 Results and Analysis

4.1 Overview
4.2 The Group as a Whole
    4.2.1 Feeling 'good' about computers
    4.2.2 Programming performance
    4.2.3 Attitude, instructor helpfulness and teaching style
    4.2.4 Learner control, progress, work difficulty and anger
    4.2.5 Daily work sheets
4.3 Lecture/Lab Contrasts
    4.3.1 Previous computer experience
    4.3.2 Previous computer programming experience
    4.3.3 Self confidence when working with computers
    4.3.4 Group allocation
    4.3.5 Preferred learning style
    4.3.6 Free choice
    4.3.7 Instructor's teaching style
    4.3.8 Was the teaching style, what you expected?
    4.3.9 Instructor helpfulness
    4.3.10 Learner control
    4.3.11 Work comprehension, progress and anger
    4.3.12 Daily attitude
    4.3.13 Programming performance
Chapter 5 Summary, Discussion and Conclusions

5.1 Context of the Study
5.1.1 Constraints
5.1.2 Field independence and male/female contrasts

5.2 Novice Programmers
5.2.1 Active student participation
5.2.2 Greater student interaction in class
5.2.3 Work sheets

5.3 The Lecture and Lab Instructional Methods
5.3.1 Lecture approach
5.3.2 Lab approach

5.4 Conclusion

References

Appendices

Appendix A. Lecture - lab class notes
Appendix B. Example sheets for the guided self-discovery group
Appendix C. Laboratory work sheets
Appendix D. CSED-420 course description
Appendix E. Group embedded figures test
Appendix F. Programming quizzes
Appendix G. Student attitudes towards computing.
Appendix H. Daily student log
Appendix I. Interview schedule
Appendix J. Interview transcripts
List of Tables

Table 1. Time required of participants in addition to the scheduled course work. 24
Table 2. Mean difference between paired groups: Attitude question 2. 34
Table 3. Correlation values for daily attitude, Q.4 and Q.8 of the log sheet. 36
Table 4. Mean difference between paired groups: Log sheet summary. 38
Table 5. Mean difference between paired groups: Log sheet, question 4. 40
Table 6. Mean difference between paired groups: Log sheet, question 7. 42
Table 7. Mean difference between paired groups: Log sheet, question 12. 46
Table 8. Mean difference between paired groups: Log sheet, question 9. 46
Table 9. Mean difference between paired groups: Attitude question 1. 49
Table 10. Lec/lab: Correlation for 'experience' and 'self-confidence'. 50
Table 11. Mean difference between paired groups: Log sheet, question 8. 55
Table 12. Mean difference between paired groups: Log sheet, question 3. 59
Table 13. Correlation values for Q.7, Q.9 and Q.12 of daily log sheet. 61
List of Figures

Figure 1. Summary of research variables. 4
Figure 2. Research design. 21
Figure 3. A summary of the class activities. 22
Figure 4. Interview selection. 28
Figure 5. Summary responses for length of course preference. 29
Figure 6. All: Student age range. 32
Figure 7. All: Previous computer experience. 33
Figure 8. All: Quiz and exam results. 35
Figure 9. All: Daily attitude, instructor help, teaching style. 37
Figure 10. All: Control, progress in class, anger, work difficulty. 41
Figure 11. All: Most valuable course materials. 43
Figure 12. Lec/lab: Group allocation. 47
Figure 13. Lec/lab: Previous computer experience. 48
Figure 14. Lec/lab: Programming experience. 49
Figure 15. Lec/lab: Reactions to group allocation. 50
Figure 16. Lec/lab: Preferred learning styles. 52
Figure 17. Lec/lab: Given free choice, what style would you opt for? 53
Figure 18. Lec/lab: Was the teaching style what you expected? 56
Figure 19. Lec/lab: Instructor helpfulness. 58
Figure 20. Lec/lab: Preferred classroom organization. 60
Figure 21. Lec/lab: Ability to comprehend the work. 62
Figure 22. Lec/lab: 'I felt I made progress in class.' 62
Figure 23. Lec/lab: Feelings of anger. 63
Figure 24. Lec/lab: Daily attitude. 64
Figure 25. Lec/lab: Programming performance 65
Figure 26. An instructional framework for novice programmers. 78
Acknowledgement

I wish to thank Dr. Stuart Donn for his help and guidance during my studies. I also wish to thank my parents, George and Isobel Clarke, for their endless support and encouragement.
Chapter 1

Introduction

Computer programming is a relatively new phenomenon. Recent technological developments have permitted a high level of user interaction with the computer. Error reporting and program output provide instant student feedback, a facility uncommon to other disciplines. Instructional methods in response to this new pedagogy have been many and varied, ranging from a very traditional curriculum orientation through to radical forms of independent student study.

1.1 The Purpose of the Study

The purpose of this study was to extend the knowledge of instructional methods used for novice programmers by examining, and contrasting the perceptions of learners to two specific teaching approaches. The first, the lecture-lab approach, clearly distinguishes between theoretical instruction and practical application, the separation of classroom instruction from 'hands-on' experience. During the lecture phase the students are introduced to programming concepts with accompanying command syntax and examples illustrating various applications. During the lab phase students move to the computer room for 'hands-on' experience to verify, consolidate and reinforce the material covered in class.

An alternative method, guided self-discovery, uses the 'hands-on' experience as the instructional technique itself. Students work at the computer virtually from the first minute they enter the classroom. The aim is to leave the students free to experiment, explore and discover, furnished with just enough information to ensure success. The interactive nature of the computer (i.e. reporting error messages and displaying program output) enables the students to develop a knowledge of the programming concepts, language syntax and semantics.
From a pedagogical perspective it would be an error to assume that the guided self-discovery method requires less attention to detail than the lecture-lab method. Both instructional methods demand that the teacher be well organized and clear about the objectives and activities for each instructional session.

1.2 The Need for the Study

The literature on instructional methods for novice programming suggests a variety of approaches, including versions of the lecture-lab and guided self-discovery methods. Much of it is largely descriptive in nature (Friedman and Koffman, 1978; Rogers, 1981; Shneiderman, 1985; Marchionini, 1985). Some researchers have sought to examine the effectiveness of different approaches by focusing on specific aspects of the learning environment, for example:

- Use of an appropriate analog or model (Mayer 1975),
- Achievement Scores (Lemos, 1975),
- Templates (Dalbey, Tourniaire & Linn, 1986) and
- Default Strategies (Pea, 1986).

In the critical interplay between teacher and learner, the learners' perception of the instructional environment is perhaps one of the most important yet least understood variables. Little research has been devoted to understanding the learners' perceptions of different instructional methods advocated by both practitioners and researchers. In the absence of such knowledge, teachers have been left with the task of selecting an instructional method based upon personal preference and prior classroom experience. This study attempts to provide an insight into the students' perception of two instructional methods, and thus extend the knowledge base for decision making about learning environments for novice programmers.

1.3 Overview of the Study

An introductory computer programming class of sixteen university students was divided into two groups. During the experimentation period one group was taught using the Lecture-lab approach and the other using the
Guided Self-discovery approach. The experimentation period covered one third of the total course time. To ensure an even balance of abilities between the two groups, student allocation was based on a test of programming ability and a test of field independence. During the study students completed two attitude questionnaires to determine any change in their attitude towards computing. They were given three tests of programming ability to track their achievement levels before, during and after the experimentation period. They were required to complete a log sheet for every class recording their reactions to the instructional approach used. Finally, eight students, four from each group, were interviewed at the conclusion of the experimentation period.

1.3.1 Hypotheses

The hypotheses stated below are based on the notion that by contrasting the Lecture-lab and Guided Self-discovery methods, the following could be said, that:

General:

1. The instructional method (i.e. lecture or guided self-discovery) has no effect on student mean achievement gain.

2. There will be no change in attitudes towards computing by the students from either group.

Student perceptions of the instructor:

3. Both groups will be satisfied with the level of help offered by the instructor.

4. Both groups will be satisfied with the organization of the classes.

Student perceptions of the instructional style:
5. Both groups will be satisfied with the style of instruction they experience.

1.3.2 Research Variables

The students were divided into two groups. There were four control variables (Figure 1). Both groups took the same introductory programming course. Both groups were taught by the same instructor. As student performance was likely to be influenced by field independence (Smith, 1987; Post, 1985) and previous programming ability, the composition of the two groups was further controlled by assigning students equally based on a field independence test (Appendix E) and the scores from a programming quiz (Appendix F).

<table>
<thead>
<tr>
<th>Control variables</th>
<th>Independent variable</th>
<th>Dependent variables</th>
</tr>
</thead>
<tbody>
<tr>
<td>2. Same instructor.</td>
<td>Level 1. Lecture-lab.</td>
<td>2. Mean Achievement Gain.</td>
</tr>
<tr>
<td>4. Previous performance.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 1. Summary of research variables.

Teaching Method, the single independent variable, had two levels: Lecture-lab and Guided Self-discovery. The dependent variables fell into three categories: attitude towards computing (Loyd and Gressard, 1985), ability based on grade scores and student perceptions of the learning environment.

1.3.3 Operational Definitions

- Lecture-lab approach: Separation of instruction into two components: a lecture followed by a laboratory practice session on the computer, includes a daily work sheet. Each component is of approximately one hour duration.
• Guided self-discovery approach: Instruction, in which direct and continuous interaction with the computer is the primary learning experience, includes a daily work sheet. Each session is of approximately two hours duration.

• Field dependence/independence: These two phrases represent a continuum along which an individual may be placed to characterize the extent to which his or her perceptions are dependent on (or independent from) cues in the environment. (Reber, 1985).

• Mean achievement gain: The difference between the grades scores on a pre- and post-test of programming ability.

• Attitudes towards computing: Student responses as elicited by a modified form of the 'Survey of Attitudes Towards Learning About and Working with Computers' questionnaire designed by Loyd and Gressard (1985)

• Student perception of the learning environment: Daily record of student reaction to the learning environment and teaching method.

1.5 Chapter Preview
An examination of the literature in chapter two identifies a variety of teaching methods advocated for novice programmers. A case is then made for choosing the Lecture-lab and Guided Self-discovery approaches for this study. Chapter three details the methodology used for the study. Specific reference is made to the instruments used to explore the students’ perceptions of the instructional environment. Chapter four is given over to the analysis of the statistical and interview data collected. The final chapter provides a summary of the results and suggests implications for instructors of novice programmers.
Chapter 2

Review of the Literature

This chapter begins with introductory comments focusing on the 'relatively undeveloped pedagogy' of computer programming (Perkins, Hancock, Hobbs, Martin & Simmons, 1986). Seven instructional approaches are then identified and the relevant research results are presented. A discussion on the need for a suitable analogy for use with novices programmers follows. The chapter concludes with a rationale for using the Lecture-lab and Guided Self-discovery approaches in this study.

2.1 A Relatively Undeveloped Pedagogy

Sleeman, Putman, Baxter and Kuspa (1986) reminded us that programming is still a new phenomenon and that researchers themselves have a limited understanding of how students learn to program, the difficulties they encounter and the misconceptions they are likely to develop. They suggested that to program effectively one must:

• Have a good knowledge of the syntax and semantics of the programming language.

• Be able to debug programs.

• Be able to analyze tasks, and design algorithms aimed at solving those tasks.

Benedict duBoulay (1986) identified five areas which present difficulties for the novice programmer. His list, which concurred in large with Sleeman et al. (1986), suggested the following areas to which student attention should be drawn:

• Orientation - finding out what programming is for, what kinds of problems can be tackled and what the eventual advantages might be of expending effort in learning the skill.
• The notational machine - difficulties associated with understanding the general properties of the machine that one is learning to control.

• The notation of various formal languages - syntax and semantics.

• Structures - plans that can be used to achieve small scale goals, such as computing a sum using a loop.

• Mastering the pragmatics of programming where a student needs to learn the skills to specify, develop, test and debug a program.

He further added that learning a programming language involved not only learning that language but also the language for managing programs, as well as the language for editing programs. Many teachers will be able to identify with his observation:

Beginners sometimes take over-dramatic action to restore the computer to a standard state, e.g. switching it off and on, abandon, or delete a complete program ....

(duBoulay, 1986, p.61)

Instructors have attempted to respond to the needs of the novice programmer by developing a variety of teaching approaches.

2.2 Instructional Approaches

Learning to program can be a very challenging and stimulating experience. Much has been written on the merit, or otherwise, of various instructional approaches used in the classroom. I have identified seven distinct approaches. Some are at opposite ends of the spectrum while others could easily complement each other. In all likelihood teachers of novice programmers use a combination of the various approaches in their classrooms. The seven approaches are:

• group interaction,
• discovery learning or hands-on,
• grammatical,
• whole program or language orientation,
• bootstrap learning,
• spiral, and
• lecture.

2.2.1 Group Interaction Approach

Two authors, Lemos (1977) and Marchionini (1984), reported improved learning outcomes by encouraging novice programmers to share, conference and discuss their work with peers. The emphasis is on students interacting with other students in the learning environment. It is an approach that has not been extensively investigated in the context of novice programming but one which Lemos (1977) and Marchionini (1984) believed promoted student achievement.

Lemos (1977) in his study, primarily designed to examine undergraduate student attitudes to programming, found that team activities such as critiquing one another's work was effective in helping students to achieve higher levels of programming proficiency. The students also reported that the group interaction reduced frustration and made the work more interesting.

More recently Marchionini (1985) found that in-class discussion of student work reinforced programming concepts, strategies and language syntax. He suggested that this was an effective way to structure the learning environment and may successfully be used in tandem with other approaches.

2.2.2 Discovery Learning or Hands-On Approach

The discovery learning or hands-on approach places strong emphasis on the interactive nature of the computer. Eighteen years ago when programming was in its infancy, Bork (1971) suggested that the 'ten finger exercise' approach to programming - whereby a student receives little or no instruction in the language before he or she
sits at the terminal for the first time - was an effective way to introduce novices to the programming
environment. He likened this discovery approach to that of a biologist learning about a strange animal by
giving it a series of selective stimuli and watching its behaviour in response to those stimuli.

In 1985, Dalbey and Linn contrasted guided-discovery with direct classroom instruction. They supported
discovery learning because it encouraged the learner to be responsible for the gathering of information and for
using feedback during the learning process. They indicated that these are the skills that students need to develop
in a programming course and believed that:

.... programming is a task which can be carried out autonomously, without constant
supervision by a teacher, and thus encourages self-initiated inquiry.
(Dalbey and Linn, 1985, p.267)

They further suggested that as computer environments provide more sophisticated feedback and error diagnosis,
they will be ideally suited to providing a discovery-learning environment which maximizes student control.

Shneiderman (1985) in a discussion of programming antecedents, concepts and outcomes, supported the idea of
discovery learning and encouraged teachers to be sensitive to the possible advantages that this particular method
has for students learning.

Some researchers warn of possible problems associated with early computer exposure . Mayer (1979) argued
that greater emphasis needed to be placed on emphasizing the concepts of transactions, prestatements, and
chunks before exposing students to the hands-on experience. While recognizing the advantages of early
exposure, Dalbey et al. (1986) cautioned that an instructional method given over wholly to discovery learning
may handicap students by not presenting them with sufficient instruction in planning skills believed to be
essential for advanced programming tasks. Similar to Mayer(1979), they feared that students may suffer from
'first day' hands-on experience. They believed that modes of instruction which over utilize on-line facilities fail to communicate the value of planning because:

- the students initial programming experiences do not require planning, and therefore the advantages of planning are not apparent to them, a situation which is likely to have dire consequences at a later stage.

and that:

- the students find the on-line experience very motivating and often prefer to be on-line interacting with the computer at the expense of making progress in solving problems.

In sum, Dalbey et al. (1986) warned that the interactive nature of the computer in a hands-on learning environment had not been well channeled towards the higher cognitive skill of planning.

Finally, Saj-Nicole and Soloway (1986) argued that a focus solely on correctness of input/output behaviour in the hands-on environment often led to poorly written programs that are difficult to debug, modify, and understand.

2.2.3 Grammatical Approach

The grammatical approach heavily emphasizes the logical step-by-step exposition of the grammar of a computer language. This approach seems to be widely used particularly by new teaches of computer programming. It is a very simple teaching approach. A skill is broken down into a series of individual steps and the whole class is taken methodically through each consecutive step. Its popularity stems from the control it permits over a class and the ease with which it can be used. It is very much teacher-centred, deliberately lock-stepping students through the instructional phase so that, in theory, everybody progresses together and in unison.
In 1971, Manzo criticized this approach. He stated that it was a common but ineffective way of teaching programming languages. He suggested that the grammatical approach appealed to teachers who were logically oriented. He felt there were dangers in using this logic as the sole basis for instruction and argued that a difference exists between a logical approach to a subject area and a pedagogical approach. In the logical approach the students take a long time to get to the point at which they can write programs, and they do not gain much of a feel for the nature of the programming art because they spend so much time on the rules of grammar. By contrast, a pedagogical approach emphasizes a way of bringing students quickly to using the language. Manzo advocated a 'whole program' approach (i.e. studying complete programs rather than isolated pieces of grammar and vocabulary) as an alternative.

In a later study, Lemos (1975) attempted to examine empirically the grammatical approach by comparing it with the whole program approach. He concluded that there was no significant difference between the two methods based on a test of programming ability.

In a more recent review, Postma and Laing (1981) argued that the grammatical approach was largely ineffective and that novice programmers should be introduced to programming with an emphasis on algorithms and by the use of the whole program approach. They argued for semantics ahead of syntax.

2.2.4 Whole Program or Language Orientation Approach

As early as 1971, Bork hinted that an important clue in learning to program might come from the experience instructors had with the audiolingual method for teaching foreign languages. He argued that the corresponding 'conversation unit' for programming is the 'whole program'. In this way it was analogous to the audiolingual method whereby students encountered complete units of conversation and not just isolated pieces of grammar and vocabulary. He felt that students should be exposed to whole programs, and then assigned tasks that involve relatively small modifications of those programs.

The results of a study by Lemos (1975) which contrasted the whole program and the grammatical approach:
... did not support Bork's subjective opinion [of 1971] that a whole program approach was superior to a grammatical approach. (Lemos, 1975, p.29)

As mentioned earlier, in the 1981 Postma and Laing argued for the adoption of a whole program approach because it emphasized algorithms and semantics ahead of grammar and syntax. This approach, they claimed, enhanced student learning.

In 1981, Manzo argued for an expansion of the language-orientation approach so that students would be first taught to listen and to talk in a computer-like language and later to read and trace programs. More or less concurrently they would observe a teacher, presumably a skilled programmer, attempting to create and debug programs. Following this, the students would then become autonomous workers involved in individual programming tasks.

Finally in 1986, Dalbey et al. conducted research designed to Assess the Cognitive Consequences of Computer Environments for Learning (ACCCEL). Their report suggested that rather than beginning programming instruction with drill and practice based on specific language features, it was quite advantageous to begin with comprehension of whole programs. Students would start with reasonably sized programs (10-15 lines of code) and then be encouraged to come to understand them by executing and modifying them.

### 2.2.5 Bootstrap Learning

Perkins et al. (1986) made a claim for what they term 'bootstrap learning'. They believed it to be an important variable in learning to program. In this approach students go significantly beyond what they have been taught, displaying significant powers of invention. They believed that not only students with a flair for programming, but even students who are not doing that well can occasionally invent tactics for themselves. They were critical of the relatively undeveloped pedagogy of computer programming instruction and argued that it fell short on two fronts. It did not teach directly the sorts of schemata a skilled programmer needs (an objection also leveled at...
discovery learning), and moreover it did not guide students in ways that foster bootstrap learning of those schemata. One consequence is that youngsters vary widely in their progress:

... succeeding only to the extent that they happen to bring with them characteristics that make them good bootstrap learners in the programming context. (Perkins et al. 1986, p.53)

The point they made is that even without carefully designed teaching strategies some students still learn. They believe that the students ability to 'pull themselves up by the bootstraps' could be usefully employed in an instructional approach which deliberately encourages student initiative guided by sound programming principles.

2.2.6 Spiral Approach - Syntax and Semantics in Parallel

Syntax first? Semantics first? This issue appears regularly, in various guises, in the literature. Some authors argue that language syntax and semantics should be taught concurrently, one supporting the other. Shneiderman (1976) suggested that semantic knowledge is acquired through meaningful learning and is resistant to forgetting while syntactic knowledge must be acquired through rote learning and, unless rehearsed frequently, is subject to forgetting. He argued for an approach where novice programmers would be taught small increments of syntactic and semantic knowledge in parallel.

In a similar vein, Marchionini (1985) argued for a 'developmental' approach to teaching programming which stresses concepts rather than syntax, but includes the two. He suggested that students should understand that the mastery of general concepts facilitates programming and problem solving, not just the mastery of the idiosyncratic nature of a particular language. He described this as a spiraled curriculum wherein students are studying, modifying and extending both new and existing programs:

Thus mid-way through a course students may be working on two assignments: modifying a complex program involving new and advanced concepts, and applying previously learned concepts by writing a complete, relatively simple program. (Marchionini, 1985, p.15)
He suggested ten levels in a developmental sequence for designing programming activities incorporating both semantic and syntactic knowledge:

- enter and execute simple programs,
- read and predict output,
- complete by supplying the missing statement,
- modify existing programs,
- extend existing programs,
- run test data,
- debug logic errors,
- design programs (algorithm),
- code from design, and
- develop and write complete programs.

Sauter (1985) concluded that students may benefit from developing logical skills and also syntactic rules in parallel. She suggested that students need help in both these areas but that it varies from student to student, from time to time. The responsibility rests with the teacher to determine what is required and when. In concurring with the other authors, she believed that a combination of the two is essential throughout the learning process.

2.2.7 Lecture Approach

The lecture approach is a common method for teaching programming to beginners. All instruction takes place in the lecture room. At the completion of the lecture, the instructor departs and the students move to a laboratory to put the theory into practice. Dalbey and Linn (1985), after reviewing a number of teaching methods, believed that this approach provided the most flexible and adaptive format for overcoming many practical constraints within schools and institutions. Hence its popularity. When they contrasted the lecture approach with a discovery learning approach they concluded that the first minimized student control of learning while the other maximized it. They further concluded that probably the two methods should be combined to ensure the coverage of all skills necessary for students to become effective programmers.
A study by White (1980) evaluated the lecture approach by comparing it with a self-paced programming approach. His results demonstrated that final exam scores were higher in the self-paced approach.

Finally, Borich (1986) argued that lectures are useful for imparting facts and rules but that if student learning is to go beyond such basics then alternative strategies need to be employed. He suggested a combination of approaches, including some of those discussed previously, to ensure meaningful learning and to promote student achievement.

That concludes the review of the seven teaching approaches. Despite the variety of methods advocated above, the researchers seemed to agree on one thing— the need for a suitable analogy or model when teaching a programming language.

2.3 The Need for a Suitable Analogy or Model

Explicitly or implicitly, researchers expressed a common concern about the way in which students perceived the link between and the programming language and the computer. The problem was presented thus:

• there is a common student misconception that the computer has 'human reasoning' capabilities (duBoulay, 1986; Pea, 1986; Putman, Sleeman & Baxter, 1986).

And to overcome this:

• students need a suitable analogy or model upon which they can 'hang' their ideas on, and therefore make sense of the operation of 'the computer' (Mayer, 1975a, 1976, 1979; Dalbey and Linn, 1985; duBoulay, 1986; Perkins et al, 1986).
What students bring with them into the classroom must influence the learning process. For Ausubel, Nowak and Hanesian (1968) the anchorage of new information to existing ideas is critical for meaningful learning. In meaningful learning the process of acquiring information results in the modification of both the newly acquired information and the relevant aspects of the cognitive structure to which the new information is linked. If specifically relevant anchoring ideas are not available then the new material is likely to be poorly assimilated giving rise to relatively unstable or ambiguous meanings, having little or no long-term retention. To overcome possible misconceptions about computers, the use of a suitable analogy or model may utilize existing information within the students' cognitive structure to promote the anchorage of new ideas.

Benedict duBoulay (1986) found that students often equated 'the operation of a computer' to that of 'human reasoning'. He warned that very often students form quite reasonable theories of how the system works given their limited experience, except that their theories are often incorrect. He believed that the notion of the system making sense of the program according to its own very rigid rules is a crucial idea for the learner to grasp. He added that teachers compound the problem by their use of anthropomorphic language:

... it was trying to ..., it thought you meant ... (duBoulay, 1986, p.61)

Such statements often suggest that the system has the normal human ability of being able to infer what is meant from what is said. In a similar discussion on student misconceptions, Pea (1986) defined three classes of bugs that he believed bedeviled novice programmers:

- Parallelism - the assumption that different lines on a program can be active.

- Intentionality - when the student attributes foresight to the program and in doing so goes beyond the information given.

- Egocentrism - when students assume that there is more of their meaning for what they want to accomplish than is actually present in the code they have written.
He concluded that these three cases of conceptual bugs are rooted in a superbug and that the default strategy is:

... that there is a hidden mind somewhere in the programming language that has intelligent interpretive powers. (Pea, 1986, p.33)

He argued that novice programmers have a predominant analogy that guides their behaviour when they write programming instructions to a computer - that of conversing with a human. The problem is that while people are intelligent interpreters of conversations, computers are not. He warned that the pervasiveness of programming misunderstandings that arise from the tacit applications of the human conversational metaphor are widespread.

Student misconceptions and the use of faulty analogies have prompted researchers to examine the use of suitable analogies or models for introductory programming courses. Analogies or models that do not involve highly technical jargon but are appropriate for the level at which novice enters the programming environment.

Mayer investigated the use of models to aid novice programmers. In 1975, he provided a diagrammatic model of a computer expressed in familiar terms. He experimented with two groups of novice programmers, one group being exposed to a model while the other was not. He concluded that an important variable in instruction was the presentation of a suitable model. In 1976 he reported another study that demonstrated prior exposure to a meaningful and familiar model, especially a concrete one, was a powerful aid in learning to program. In 1979 his research indicated that the full range of instructional sequences in computer programming was not being exploited by teachers. In particular he highlighted the transactional level - the general functions of the computer that underlie programming statements:

Transactions provide a means, for the novice, of explaining what is going on inside the computer when a particular statement is executed and of relating the new technical language, that he or she must learn, to general operations, locations and objects that he or she is already familiar with. (Mayer, 1979, p.4)
This was consistent with notion presented by Ausubel et al. (1968), described above. Mayer (1979) went on to suggest that teaching programming did not require an understanding of machine-level hardware and operations. In his studies he found analogies such as a 'ticket window' or an 'eraseable scoreboard' to be quite suitable.

Dalbey and Linn (1985), in a literature review of the demands and requirements of programming, reported that several researchers were studying the ways in which instruction can build on the intuitive mental models that novices bring to the learning environment. They suggested that a programming language will be easier to learn if its structure corresponds to the way novices naively try to formulate process descriptions.

Benedict duBoulay (1986) saw the concept of an analogy or model as an important factor in teaching novice programmers. He felt there was a need to:

... present the beginner with some model or description of the machine she or he is learning to operate via the given programming language. It is then possible to relate some of the troublesome hidden side-effects to events happening in this model, as it is [these] which often cause trouble for beginners. (duBoulay, 1986, p.71)

duBoulay (1968) also warned of the danger of a poor analogy. He cited the example of the widespread use of 'a box with a label on it' to describe the concept of a variable. To forestall possible student confusion, it is important to stress that each box can hold only one item at a time.

Finally while no consensus exists on which analogy or model is best, if teachers are to use an analogy or model then they should have at their disposal one which is neither ambiguous nor credits the machine with powers greater than students are likely to perceive it as having. The analogy or model should be carefully selected, and teachers need to be constantly aware of student misconceptions that may arise from its use (duBoulay, 1986; Pea, 1986; Putman et al. 1986).
2.4 Conclusion

There are clearly a number of ways to approach the teaching of programming. The advice offered by Dalbey and Linn (1985) is worthy of serious consideration:

The implications of this research are that individual differences may be a significant factor in what is learned from instruction. Perhaps it may be important to design instruction for a variety of learning styles. (Dalbey and Linn, 1885, p.267)

The concept of being able to provide multiple learning environments catering for a variety of learning styles may appear to be impractical, if not well nigh impossible. How then might this review of instructional methods be used to explore student perceptions? Fortunately, a dichotomy of teaching approaches is possible. Upon closer inspection, the approaches may be classified as being either teacher centred or student centred. Borich (1986) uses a similar classification in his discussion of general teaching styles. Of the seven teaching methods reviewed:

- the grammatical approach,
- the spiral approach, and
- the lecture approach.

may be classified as being teacher centred. The focus of instruction is clearly on the teacher. With minor shifts in instructional emphasis, a teaching style could easily be developed to encompass the main features of all three. I will use the general label Lecture-lab to refer to this new teacher centred approach. It will be an approach which has its emphasis on formal classroom instruction. The primary objective is to introduce the student, via a lecture style environment, to programming concepts with accompanying command syntax. Examples illustrating various programming applications would also be discussed. By contrast, the three approaches:

- the group interaction approach,
• the discovery learning or hands-on approach, and
• bootstrap learning

may be classified as being student centred. Again with minor shifts in instructional emphasis, these three can be synthesized into a single teaching approach. I will use the general label Guided Self-discovery to refer to this new student centred teaching style. This approach will use the 'hands-on' experience as the primary form of instruction. Students will be free to experiment, explore and discover, furnished with just enough information to ensure success.

The features of the seventh teaching method, the whole program or language orientation approach do not necessarily exclude it from either the teacher or student centred orientations. As such, it is feasible for it to be used judiciously in both the Lecture-lab and Guided Self-discovery environments.

The teacher or student centred classification enables some sense to be made of what appears to be a disparate number of teaching approaches. It provides a focus for examining student perceptions. The study described in the following chapters used this classification, and the composite approaches labeled Lecture-lab and Guided Self-discovery, to examine student perceptions within the learning environment. In acknowledging the need for a suitable analogy or model, the 'erasable chalkboard' was chosen as one of the better analogies and was used throughout this study.
Chapter 3

Methodology

3.1 The Course

Computers For Instruction (CSED 420), a 1988 summer school credit course offered at the University of British Columbia, was chosen as the introductory computer course for the project. The course was a three week intensive introduction to BASIC programming. The same course is available over a thirteen week semester in the regular fall and winter sessions at the university. The research project was conducted during the second week of the course. In the first week the students were introduced to the computer and simple programming commands. They became familiar with entering, editing and executing simple programs. They learnt to save and retrieve programs from disk. They also received instruction in printing formats, the use of library functions and looping structures. Having dealt with many of the 'house-keeping' and introductory activities in week one, week two, was devoted entirely to programming in BASIC. During the project period students studied:

- conditional and unconditional statements,
- the random number function,
- single dimension arrays,
- computer graphics, and
- subroutines.

In the third week the students returned to the regular lectures and were given instruction in sorting, searching and file manipulation. They were also introduced to educational software applications and the principles of software evaluation.

Participation in the study was voluntary. Students who choose not to participate in the project were free to attend either of the instructional methods without having to complete any questionnaires or the log book reports. Those participating were asked to commit a up to 135 minutes to the project in

21
addition to the scheduled course work. A breakdown of the student time commitment for data collection is given in Table I.

Table 1. Time required of participants in addition to the scheduled course work.

<table>
<thead>
<tr>
<th>Activity</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Field Dep./Indep. Test</td>
<td>15 minutes</td>
</tr>
<tr>
<td>Initial Questionnaire</td>
<td>15 minutes</td>
</tr>
<tr>
<td>Final Questionnaire</td>
<td>10 minutes</td>
</tr>
<tr>
<td>Log Book Recording</td>
<td>50 minutes (5 x 10 minutes)</td>
</tr>
<tr>
<td>Interviews (8 students only)</td>
<td>45 minutes</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>135 minutes</strong></td>
</tr>
</tbody>
</table>

All information collected during the study remained confidential and was not analyzed until the final course marks for all students were submitted to the university student records office.

3.2 The Instructors

Two instructors were used for the course. One was responsible for teaching during the first and third weeks of the course. This instructor used a Lecture-lab approach. During the project period when the students were divided into their two groups a second instructor was responsible for all instructional sessions and employed the appropriate teaching style (i.e. Lecture-lab or Guided Self-discovery).

Instructional sessions were of one hour duration. Each instructional session was followed by a scheduled laboratory session. The laboratory sessions were supervised by a teaching assistant who was available to answer student enquiries. During the first and third weeks of the course, the instructor periodically visited the laboratory to answer questions. During the project period the instructor was present during each laboratory session to monitor student progress.
3.3 Measuring Instruments

The Group Embedded Figures Test (Witkin, Oltman, Raskin and Karp, 1971) was used to place students along a continuum which characterized them as being either more field independent or more field dependent (see Appendix E for instrument).

Ability based on grade scores was used to determine student performance. Two programming quizzes given to the students, one administered before and the other after the experimentation period. These quizzes were similar to those used for many years at the university and found to be reliable indicators of student performance in computer science education. The students also sat a final examination at the end of the course. See Appendix F for test instruments.

To assess student attitudes towards computing, and to determine any change that may occur as a result of the two teaching methods, a slightly modified form of the questionnaire 'Survey of Attitudes Towards Learning About and Working with Computers' (Loyd and Gressard, 1985) was administered immediately before and after the experimentation period. See Appendix G for instrument.

The student perceptions of the learning environment were captured by a daily student log-book specifically designed for this project. A modified form of the 'A-State Scale' (Vavrik, 1987) allowed students to rate their reactions to twelve aspects of the learning environment. In addition, students were asked to record what they felt was successfully learnt in each session, the amount of time they spent each day on computing, and any other reactions they might have to the learning environment. Students completed daily log sheets before, during and after the project period. See Appendix H for instrument.

Finally eight students (two randomly selected from each of the four cells) were interviewed after the project period. The interview schedule is contained in Appendix I.
3.4 Research Design

Prior to the experimentation period, participants were given a field independence test (Group Embedded Figures Test, Appendix E) and a programming quiz (a regular component of the course work, Appendix F). Students were assigned to one of the two teaching methods (Lecture-lab or Guided Self-discovery) and evenly blocked based on their field independence and test scores (Figure 2).

<table>
<thead>
<tr>
<th></th>
<th>Lecture-lab</th>
<th>Guided Self-discovery</th>
</tr>
</thead>
<tbody>
<tr>
<td>Field Independence</td>
<td>Students</td>
<td>Students</td>
</tr>
<tr>
<td>Field Dependence</td>
<td>Students</td>
<td>Students</td>
</tr>
</tbody>
</table>

Figure 2. Research Design.

3.5 Instructional Methods

Class work for each day was divided into three parts (Figure 3):

1. The introduction, which detailed the material to be covered in the lesson.

2. The instructional component, which dealt with new concepts and programming instructions.

3. The laboratory session, where students consolidated the day's activities at a computer and attempted a laboratory work sheet.
Both the Lecture-lab and Guided Self-discovery groups were given the same introduction which briefly detailed the content of the day's lesson.

The curricula covered in the instructional component for both groups was identical. The Lecture-lab group received instruction in a lecture theatre. The instructor would briefly list and describe the new programming concepts and statements and then work through a set of examples to illustrate the use of these in a BASIC program. See Appendix A for the lesson preparation notes that were used for the Lecture-lab classes.

By contrast, the Guided Self-discovery group were given no direct instruction. Instead they were given the same examples but on paper. These examples included a printout of the program listing and execution. Where the nature of a concept or statement was particularly obscure and not likely to be reasonably intuited from the example, it was supplemented with a written comment. The Guided Self-discovery students also were given disk copies of each example so that they could, if they wished, load and run the examples at their own computer. See Appendix B for a copy of the programming examples sheets used by the Guided Self-discovery group.

<table>
<thead>
<tr>
<th></th>
<th>Lecture-lab</th>
<th>Guided Self-discovery</th>
</tr>
</thead>
<tbody>
<tr>
<td>Introduction</td>
<td>Common introduction for both groups</td>
<td></td>
</tr>
<tr>
<td>Instructional component</td>
<td>• Instructor describes concepts and statements.</td>
<td>• Programming example sheets distributed.</td>
</tr>
<tr>
<td></td>
<td>• Students and instructor work through examples.</td>
<td>• Software copies of programming examples available.</td>
</tr>
<tr>
<td>Laboratory</td>
<td>Common work sheets for both groups.</td>
<td></td>
</tr>
</tbody>
</table>

Figure 3. A summary of the class activities.
Both groups were given identical laboratory work sheets. There was a work sheet to coincide with each day's instruction. Each sheet contained about five to six problems and were designed so all students could complete at least two to three problems within the scheduled laboratory session. Refer to Appendix C for a copy of the work sheets.

3.6 Data Collection Procedures

During the first week of the course all students attended regular lecture sessions, each of approximately one hour duration. These sessions were followed by laboratory sessions where students put into practice the theory covered during the lecture. A teaching assistant was available during the laboratory sessions to answer student questions. A number of activities related to the project were also conducted during the first week. These are described below.

With the exception of the programming quizzes and the field independence test, all data collected was sealed and stored until the final students marks for the course had been submitted to the student records office. When collecting information from the students, they were given individual envelopes into which the information (e.g. daily log sheet) was placed and sealed before being handed into the instructor.

On the first day of the course the students were given a student information booklet containing information about the project and an invitation to participate. The booklet also contained a consent form. Students were given the opportunity to ask questions about the project. On day two students returned the consent forms and sixteen students volunteered for the project (seventeen students were on the courses). Each student was given a set of daily log sheets which were to be completed for three days prior to the project period, for the week during the project week and then for a further three days after the project. On consecutive days immediately before the project week the students were given the field independence test (Appendix E), the first programming quiz (Appendix F) and the attitude questionnaire (Appendix G).
Based on the field independence test and the programming quiz results, the students were allocated to either the lecture-lab or the guided self-discovery group. A different instructor was used for the project week. This instructor was responsible for both groups and therefore the instructional times were staggered to avoid clashes. The students were asked not to discuss the sessions or to share materials between the two groups. The guided self-discovery group received were the first to receive their instruction each day, followed immediately by a scheduled laboratory session. Later in the day the lecture group were given their instruction, followed immediately by a laboratory session.

Both the lecture and guided self-discovery groups were given a common introduction to the day's activities. This included a listing of the concepts or instructions to be learnt for that day and any administrative matters that required attention. The lecture group were given a formal lecture. The preparation notes used for the lecture sessions are contained in Appendix A. Instruction for the lecture group was of two forms: verbal and visual. The instructor spoke about the various programming instructions and used the chalkboard to note points of importance and work through several programming examples. The guided self-discovery group were given a series of handouts which contained the same examples as those written on the chalkboard for the lecture group (Appendix B). These sheets contained the absolute minimum amount of instructional information and students were asked to read through the examples carefully and to try to work out the effect of the individual instructions and how the programs worked. To avoid unnecessary typing the guided self-discovery group were given software copies of all the programming examples.

Both groups were encouraged to ask questions. Direct solutions were not given to the students. Whenever possible the students were directed to the programming examples for ideas and solutions.

Throughout the project period the daily log sheets were completed and collected from the students. On the last day of the project the students were given the attitude questionnaire again (Appendix G). On the day following the project period the students were given the second programming quiz (Appendix F). All eight interviews were conducted within two days of the completion of project period (Appendices I and J).
After the project period the students returned to regular lecture sessions with the original instructor. On the last day of the course the students sat the final examination (Appendix F).

3.7 Data Analysis

The data analysis focused specifically on:

- the group as a whole (sixteen students),
- the lecture-lab sub-group (8 students) and
- the guided self-discovery sub-group (8 students).

For ease of reference the lecture-lab group will be referred to simply as 'lecture group' and the guided self-discovery group will be called the 'lab group' for the remainder of the analysis and discussion.

While a detailed analysis of the field dependent/independent and male/female groups is beyond the scope of this study, data for these groups has been included in the statistical tables and interview summaries and will referred to in the final chapter.

An explanation of the analysis of the data, and examples using actual tables and figures from the study follows.

3.7.1 Interview analysis

Eight students were interviewed, four from each of the lecture and lab groups (two selected randomly from each of the four sub-groups (see Figure 4). All eight interviews were fully transcribed. The interview schedule is contained in Appendix I. The interview data appears in two formats throughout this analysis. The first is in the form of summary tables briefly capturing each interviewee response to a particular question. An example of this format is given in Figure 5.
The second is in the form of direct quotations from the original transcripts.

3.7.2 Statistical Analysis

T-tests were used to determine the statistically significant differences, if any, that occurred between the lecture-lab and guided self-discovery groups before, during and after the project period. They were used to compare the differences between the mean results for the two groups on their responses to the daily log sheet, the attitude questionnaire and the quiz results. Unpaired t-tests enabled a direct comparison of changes or differences between the two groups. Paired t-tests enabled a direct comparison of changes or differences within the groups over the three time intervals i.e. before, during and after the project period.
The t-tests on the daily log sheet included a test for the average of the student ratings for the twelve items and also a t-test for each of the twelve individual items. The t-tests for the attitude questionnaire included a test for the average of the student ratings for the forty items and also a t-test for each of the forty individual items. The t-tests on the quiz results used the final marks for each of the tests.

While the statistical analysis produced many tables, only those in which 15% of the comparisons were statistically significant were retained for closer examination. Only in these tables were the differences between, or within, the groups felt to be more than just random events. For example, a table which had 30 comparisons must have had at least five which were statistically significant ($p \leq .05$) for it to be considered for further analysis.

In some tables using the paired t-tests, the results given for the same time intervals (i.e. before, during and after) differed. This occurred because of the nature of the pairing process required to calculate this statistic. For example, on occasions a student may have fully completed the 'before' and 'during' log sheets but only completed eighty percent of the 'after' log sheet. This meant that when pairing the 'before to during' and the 'before to after' time intervals, the scores for 'before' statistic were different in the two intervals, i.e. there was a one hundred percent pairing in the first instance and only an eighty percent pairing in the second instance resulting in a difference in the 'before' statistic that was calculated for the two intervals.

Pearson's correlation coefficient was also calculated to determine the degree and direction of the relationship between the questionnaire items, demographic information and quiz scores collected during the project period.
Chapter 4

Results and Analysis

4.1 Overview

The analysis of the interview and statistical data is described under the following headings:

- the group as a whole, and
- the lecture/lab contrasts.

It concludes with a brief summary of the salient points. Please note that two complete interview transcripts are contained in Appendix J.

4.2 The Group as a Whole

Sixteen students volunteered to participate in the project. Seven were female and nine male. Their diverse academic backgrounds provided the project with a wide representative sample of the student population. Student backgrounds included physical education, mathematics, biology, library, physics, art, geology and industrial education.

Figure 6 gives the age spread of the students undertaking CSED-420. The majority of the students were between the age of twenty-three to forty (Figure 6). Eleven were completing their fifth year of university study to obtain or update their teaching degree. One student was in second year, one in third year, and two in the fourth year of their education programme. One student was beginning a masters programme in education. Based on my experience with the computer studies classes over the two previous semesters, this age spread was typical of students undertaking the course.
The data collected on previous computer experience, defined in the broadest sense of the word (which included word processing, computer programming, book keeping, computerized measuring instruments used in a laboratory, and library cataloguing using computers), showed that the majority of students had some previous experience with computers, ranging from less than one week to more than one year (Figure 7).

Because prior experience could conceivably distort the analysis of the lecture and lab group data, and affect the interpretation of the project results, a Pearson's correlation matrix was computed to determine the relationship, if any, between prior experience and student performance. In sum, based on the data collected from the participants in this project, no significant statistical relationship between previous computer experience and student performance on the university programming tests was found.

4.2.1 Feeling 'good' about computers

One item on the attitude questionnaire asked students to report how 'good' they felt with computers. The definition of the word 'good' was left up to the individual student. After analyzing the data and from informal
conversations during the project with the students, feeling 'good' about computers was equated with having achieved some measure of success with 'the computer' during the scheduled class and laboratory times.

![Figure 7. Previous computer experience (n=16).](image)

This was as simple as using editing commands in the first few days through to manipulating screen formats later on in the course. Often success was related to the number of problems they were able to complete on the daily work sheets. Students were disappointed if they went beyond scheduled class and laboratory times just to cope with the daily tasks assigned to them (i.e. two to three problems from the work sheets). On the other hand, if the students achieved the daily tasks within the scheduled laboratory time, they were motivated to work in their own time on additional problems.

Consider row one in Table 2. The student rating for this questionnaire item shows a statistically significant positive change between the responses given before the project as opposed to those given after the project period.
That the students did feel better is a reflection of the success of the course and the aims as outlined in the course description (Appendix D). The above result is also consistent with the interview data in which all eight students reported feelings of success at the end of the project period. In the following interview excerpts the students indicate some of the criteria by which they measured that success:

(Lec = lecture group, lab = lab. group)

Student 1 (Lec): Being able to jot programs down as you are going.

and,

Student 4 (Lec): The ability to be able to use the computer and design programs to do what I want to do.

and,

Student 6 (Lab): Doing the work sheets and the quizzes.
4.2.2 Programming performance

Student success on the programming tests (Appendix F) is graphed in Figure 8. The mean for each of the tests is marked by a circular dot. The standard deviation of the test scores is indicated by the length of the bars extending above and below the dot.

![Mean and Standard deviation bars for Quiz 1, Quiz 2 and Exam.](image)

Figure 8. All: Quiz and Exam Results (n=16).

The first quiz tested syntax, simple program reading and had a few easy programming questions drawn from the work sheets. The second quiz had more advanced syntax, program reading and program writing problems. The final exam had a substantial section devoted to program reading and interpretation in addition to five complex program writing problems. While there was a decline in the scores over the three tests, at the conclusion of the project period (i.e. quiz 2) the mean for the group was 77.8 and the standard deviation was 12.2. This result indicated that the group, as a whole, were successful on the written BASIC programming tests and supports the data analysis in the previous section.
4.2.3 Attitude, instructor helpfulness and teaching style

'Daily attitude' is the average score of student ratings for the twelve questionnaire items from the daily log sheet (Appendix E). 'Instructor helpfulness' comes from question 4, and 'teaching style' from question 8:

\[\begin{array}{ll}
Q.4 & \text{I felt the instructor was helpful.} \\
Q.8 & \text{I was happy with the instructors teaching style.}
\end{array}\]

These three are being examined together because:

- a significant correlation was found between them, and
- the student ratings for all three showed a statistically significant positive change going into the project period and then a significant statistical negative change coming out of the project period.

Table 3 gives the Pearson's correlation coefficients for the three items. A significant correlation was found between all three for each of the time intervals: before, during and after the project period (alpha <= .05). If students are happy with the teaching style and the amount of help the instructor provides then their overall attitude to the class appeared to be positive.

<table>
<thead>
<tr>
<th>Items</th>
<th>Before</th>
<th>During</th>
<th>After</th>
</tr>
</thead>
<tbody>
<tr>
<td>Daily Attitude &amp; Q.4</td>
<td>.826</td>
<td>.9</td>
<td>.815</td>
</tr>
<tr>
<td>Q.4 &amp; Q.8</td>
<td>.829</td>
<td>.799</td>
<td>.612</td>
</tr>
<tr>
<td>Daily Attitude &amp; Q.8</td>
<td>.776</td>
<td>.857</td>
<td>.674</td>
</tr>
</tbody>
</table>

Figure 9 graphically represents the positive and negative reactions the students had before, during and after the project for each of the three items.
Consider firstly, the daily attitude scores (i.e. the average of the student ratings for the twelve items on the daily log sheet). These scores reflect the student feelings at the end of each instructional session and capture, in a very immediate sense, student reaction to the learning environment just experienced. Table 4 contains the statistical data for the daily attitude scores. Figure 9 graphically represents the changes over the three time intervals. The ratings in columns four and seven of row one of Table 4 show a significant statistical positive change when moving into the project period ($p=.04^*$) and then the reverse as the students moved back to the regular lectures ($p=.02^*$). The possible causes for these marked changes are discussed more fully as later analyses focus specifically on individual questionnaire items and interview data. Suffice to say, that the students perceive something very different about the learning environment of the project period as compared to that of the regular classes. The indication, as determined by the questionnaire ratings, is that they preferred the instruction during
the project period. This issue, and possible interpretations, is discussed more fully in the context of the project in chapter five.

Table 4. Mean Daily Attitude Rating Difference Between Paired Groups

<table>
<thead>
<tr>
<th>Groups</th>
<th>Before - During</th>
<th>During - After</th>
<th>Before - After</th>
</tr>
</thead>
<tbody>
<tr>
<td>All (n=16)</td>
<td>44.714 48.786 p=.04 *</td>
<td>48.154 40.077 p=.02 *</td>
<td>44.667 41 p=.11</td>
</tr>
<tr>
<td>LEC (n=8)</td>
<td>45.714 49.714 p=.14</td>
<td>49.875 39.625 p=.03*</td>
<td>45.714 41.143 p=.21</td>
</tr>
<tr>
<td>LAB (n=8)</td>
<td>43.714 47.857 p=.21</td>
<td>45.4 40.8 p=.42</td>
<td>43.2 40.8 p=.4</td>
</tr>
<tr>
<td>FI (n=9)</td>
<td>42 47.571 p=.06</td>
<td>46.714 36.571 p=.04 *</td>
<td>41.167 37.833 p=.33</td>
</tr>
<tr>
<td>FD (n=7)</td>
<td>47.429 50 p=.38</td>
<td>49.837 44.167 p=.28</td>
<td>48.167 44.167 p=.26</td>
</tr>
<tr>
<td>Male (n=9)</td>
<td>45.556 50.222 p=.01 *</td>
<td>49.143 42.846 p=.23</td>
<td>45.714 44.286 p=.66</td>
</tr>
<tr>
<td>Female (n=7)</td>
<td>43.2 46.2 p=.56</td>
<td>47 35.167 p=.049 *</td>
<td>43.2 36.4 p=.03 *</td>
</tr>
</tbody>
</table>

[ALL=whole group, LEC=lecture group, LAB=lab group, FD=field dep, FI=field indep, *p<.05, **p<.01]

The second of the three important changes was the student reaction to the instructor's teaching style. In Table 11, the student ratings in row one, columns four and seven show a statistically significant difference in the assessment of the instructor's teaching style during the project as compared to before or after it. It shows a statistically significant positive change (p=.007**) moving into the project period and then a statistically significant negative change as students moved back to regular classes (p=.004**). Figure 9 presents the same information but in a graphical form. In short, the students felt happier with the instructor's teaching style during the project than for either before or after the project period. The data collected during the interviews suggested that this feeling may be attributed to a number of factors. Group interaction and teacher rapport, two
which were frequently mentioned, are highlighted in the following interview excerpt:

(Lec = lecture group, Lab = lab group)

Interviewer: In the lecture group was it what you expected it to be?

Student 2 (Lec): ... (uncertain pause) ... Yes.

Interviewer: Do you want to add anything to that?

Student 2: Yes, there was more interaction [during the project lecture sessions] than there have been with our normal instructor.

Interviewer: Interaction with the instructor?

Student 2: Student interaction.

Interviewer: Ah, right . . .

Student 2: But that is a rapport, that is a part of the rapport that [the project lecturer] built with the group.

The instructor for the project period was different to the regular class instructor. The project period instructor took both the lecture and lab groups for the duration of the project period. This may be one factor which contributed to the students feeling happier about the project lecturer's teaching style. The issue of student interaction comes out even more forcefully in later analyses.

The third statistically significant student reaction is related to 'instructor helpfulness'. Figure 9 graphically represents the information from Table 5, columns four and seven of row one. The instructor was rated as being
more helpful during the project period as opposed to either before (p=.027*) or after (p=.048*) it. The students felt the instructor during the project period was more helpful than the instructor for the regular lectures.

Table 5. Mean Student Rating Difference Between Paired Groups: Log Sheet Question 4.

<table>
<thead>
<tr>
<th>Groups</th>
<th>Before - During</th>
<th>During - After</th>
<th>Before - After</th>
</tr>
</thead>
<tbody>
<tr>
<td>All (n=16)</td>
<td>3.657 4.179 p=.027*</td>
<td>4.131 3.408 p=.048*</td>
<td>3.683 3.525 p=.48</td>
</tr>
<tr>
<td>LEC (n=8)</td>
<td>3.786 4.343 p=.06</td>
<td>4.362 3.412 p=.04*</td>
<td>3.786 3.614 p=.52</td>
</tr>
<tr>
<td>LAB (n=8)</td>
<td>3.529 4.014 p=.23</td>
<td>3.76 3.4 p=.6</td>
<td>3.54 3.4 p=.76</td>
</tr>
<tr>
<td>FI (n=9)</td>
<td>3.257 3.986 p=.04*</td>
<td>4.014 3.1 p=.12</td>
<td>3.3 3.283 p=.96</td>
</tr>
<tr>
<td>Male (n=9)</td>
<td>3.644 4.333 p=.002**</td>
<td>4.243 3.8 p=.35</td>
<td>3.686 3.8 p=.72</td>
</tr>
<tr>
<td>Female (n=7)</td>
<td>3.68 3.9 p=.7</td>
<td>4 2.95 p=.09</td>
<td>3.68 3.14 p=.08</td>
</tr>
</tbody>
</table>

[ALL=whole group, LEC=lecture group, LAB=lab group, FD=field dep, FI=field indep,*p<.05,**p<.01]

The group as a whole perceived the project period to be more satisfying than the regular lectures. In the following chapter the context of the project within the actual course will address possible reasons for this result (e.g. novelty factor, change in instructor etc.).

4.2.4 Learner control, progress, work difficulty, anger

Responses to four other daily log sheet questionnaire items exhibited a statistically significant negative reaction as students moved from the project back to regular lectures. Figure 10 graphically represents the negative trend for each of these questions:
Q.3 I felt I was in control of my own learning.  
Q.7 I felt I made very little progress.  
Q.9 I felt angry.  
Q.12 I felt the work was difficult to comprehend.

A test for correlation did show a positive relationship between the four items although the correlation was neither strong nor statistically significant.

Table 12 gives the student ratings for question three. In row one, column seven, there is a statistically significant difference in the student ratings during the project period as opposed to the regular lectures (p=.02*).

The students indicated that they felt more in control of their own learning during the project period than after it. This trend is explored more fully in the analysis of the lecture/lab contrasts.

Figure 10. Changes in Learner control, Progress, Anger, Work Difficulty ratings (n=16).
Table 6 gives the student ratings for question seven. The students felt they made more progress during the project as opposed to after it. Consideration of row one, column seven, shows a statistically significant decline in the student response for the 'during to after' time interval (p=.04*). The issue of 'progress' not only received attention in the above student analysis but was a key issue for all students throughout the course and was related to the daily work sheets.

### Table 6. Mean Student Rating Difference Between Paired Groups: Log Sheet, Question 7.

<table>
<thead>
<tr>
<th>Groups</th>
<th>Before - During</th>
<th>During - After</th>
<th>Before - After</th>
</tr>
</thead>
<tbody>
<tr>
<td>All (n=16)</td>
<td>3.507 3.864 p=.15</td>
<td>3.815 3.1 p=.04 *</td>
<td>3.55 3.108 p=.16</td>
</tr>
<tr>
<td>LEC (n=8)</td>
<td>3.529 3.914 p=.16</td>
<td>3.962 3.138 p=.01 *</td>
<td>3.529 3.157 p=.34</td>
</tr>
<tr>
<td>FI (n=9)</td>
<td>3.6 3.814 p=.37</td>
<td>3.743 2.729 p=.03 *</td>
<td>3.45 2.683 p=.17</td>
</tr>
<tr>
<td>FD (n=7)</td>
<td>3.414 3.914 p=.29</td>
<td>3.9 3.533 p=.49</td>
<td>3.65 3.533 p=.74</td>
</tr>
<tr>
<td>Male (n=7)</td>
<td>3.456 3.978 p=.05</td>
<td>3.857 3.571 p=.3</td>
<td>3.514 3.571 p=.86</td>
</tr>
<tr>
<td>Female (n=7)</td>
<td>3.6 3.66 p=.92</td>
<td>3.767 2.55 p=.08</td>
<td>3.6 2.46 p=.04 *</td>
</tr>
</tbody>
</table>

[ALL=whole group, LEC=lecture group, LAB=lab group, FD=field dep, FI=field indep,*p<.05,**p<.01]

Similar trends are discussed for question 9 (p=.008**, table 7) and question 12 (p=.03*, table 8) in the following section.

### 4.2.5 Daily work sheets

Figure 11 lists the 'the most valuable resource materials' as identified by the interviewees.
Figure 11. Most valuable course materials (S=Student).

From student interviews and informal classroom conversations, 'progress in class' was viewed as the ability to successfully complete the assigned problems on the work sheets.

The following interview excerpts highlight this and other aspects of the work sheets:

(Student 6 (Lab): The work sheets were good, really helpful. You could do them, you could have success with them.

and,

Student 5 (Lab): I learnt the most from the work sheets.

and,

Student 8 (Lab): Very, useful, especially the assignments [work sheets] were according to the materials, you were not asked something without the right materials.)
Student 7 (Lab): The work sheets, they were quite good because they were enough work to keep you busy for the time you had and not excessive so that you had to stay overtime.

and,

Student 4 (Lec): So I think the way I would prefer to learn this is just total laboratory session with asking questions of people as we go along and work sheets.

While the work sheets received the greatest emphasis, students did comment on other materials, for example:

(Lec = lecture group, lab = lab group)

Student 5: The handout sheets were a good idea. The example disk was a great idea.

Interviewee: In what way?

Student 5: I would, um, copy a program from it, and then work on it, change things and play around with it.

In general though, the work sheets provided a focus for student progress in combination with either the text or programming example sheets. Furthermore, some students suggested that the work sheets could have been complemented with a solution set of exemplary programs against which they could check their programming solutions, for example:

(Lec = lecture group, Lab = lab group)

Student 2 (Lec): I would have liked] some place to check for accuracy, a check for a more efficient methods of doing something.

and,

Student 5 (Lab): Well, with the GOSUBS, I didn't realize exactly how they worked. I was getting the answers. I was getting the right outputs, but the way I was doing it wasn't maybe the best, most elegant way to do it. Needed more checking to see . . .
A lot of exercises I modified on my own, like if you did just the basic exercises you didn't learn a whole lot.

It should also be noted that the work sheets were used much more than anticipated. Recall that the students were only expected to complete approximately half of the problems on each sheet. The interviews and classroom monitoring revealed that the students diligently worked through each and every problem, which in some cases meant additional time beyond the scheduled class and laboratory times. The additional time devoted to the work sheets was of concern to the instructors. In general though, the students did not report the workload as being unbearable. Of the eight students interviewed, seven felt the workload to be reasonable and one described it as heavy.

In sum, the work sheets were important for the students to gauge their daily progress and success in class. Difficulty with the work sheets highlighted other reactions. Table 7 gives the daily students' rating of their ability to comprehend the class work. The response shows a statistically significant decline (p=.008**) as students moved from the project period back to the regular lectures. This item is consistent with reduced student control over the learning environment and feelings of reduced progress in class, as analyzed earlier.

It should also be noted that work difficulty increased as the course progressed. With the increasing difficulty of the class work and a feeling of 'loss of control' over the learning environment, it was not unexpected that the students might indicate less progress in class. Furthermore, only one general programming work sheet was distributed to the students following the project period. In the light of the analysis which found that students favoured the work sheets, the sudden withdrawal of the work sheets may have been a factor in the students declining comprehension of the daily class work. Some students did express a regret that further work sheets were not available. This may also have been a contributing factor to the increased feelings of anger reported by the students and captured in Table 8.
Table 7. Mean Student Rating Difference Between Paired Groups: Daily Log Sheet, Q.12.

<table>
<thead>
<tr>
<th>Groups</th>
<th>Before - During</th>
<th>During - After</th>
<th>Before - After</th>
</tr>
</thead>
<tbody>
<tr>
<td>LEC (n=8)</td>
<td>3.671 4.186 p=.049 *</td>
<td>4.225 3.125 p=.02 *</td>
<td>3.671 3.286 p=.27</td>
</tr>
<tr>
<td>LAB (n=8)</td>
<td>4.014 4.1 p=.8</td>
<td>3.94 3.2 p=.27</td>
<td>3.68 3.2 p=.4</td>
</tr>
<tr>
<td>FI (n=9)</td>
<td>3.386 4.043 p=.04 *</td>
<td>3.971 2.957 p=.05 *</td>
<td>3.117 3.117 -</td>
</tr>
<tr>
<td>Female (n=7)</td>
<td>3.54 3.96 p=.45</td>
<td>4.05 2.617 p=.02 *</td>
<td>3.54 2.74 p=.11</td>
</tr>
</tbody>
</table>

[ALL=whole group, LEC=lecture group, LAB=lab group, FD=field dep, FI=field indep,*p<.05,**p<.01]

Table 8. Mean Student Rating Difference Between Paired Groups: Log Sheet, Q.9.

<table>
<thead>
<tr>
<th>Groups</th>
<th>Before - During</th>
<th>During - After</th>
<th>Before - After</th>
</tr>
</thead>
<tbody>
<tr>
<td>All (n=16)</td>
<td>4.221 4.379 p=.28</td>
<td>4.315 3.638 p=.03 *</td>
<td>4.158 3.775 p=.17</td>
</tr>
<tr>
<td>LEC (n=8)</td>
<td>4.286 4.514 p=.15</td>
<td>4.55 3.475 p=.03 *</td>
<td>4.286 3.686 p=.11</td>
</tr>
<tr>
<td>LAB (n=8)</td>
<td>4.157 4.243 p=.74</td>
<td>3.94 3.9 p=.77</td>
<td>3.98 3.9 p=.82</td>
</tr>
<tr>
<td>FI (n=9)</td>
<td>3.843 4.329 p=.02 *</td>
<td>4.3 3.314 p=.07</td>
<td>3.733 3.533 p=.63</td>
</tr>
<tr>
<td>FD (n=7)</td>
<td>4.6 4.429 p=.34</td>
<td>4.333 4.017 p=.33</td>
<td>4.583 4.017 p=.08</td>
</tr>
<tr>
<td>Male (n=9)</td>
<td>4.244 4.511 p=.04 *</td>
<td>4.371 3.986 p=.17</td>
<td>4.143 3.986 p=.64</td>
</tr>
<tr>
<td>Female (n=7)</td>
<td>4.18 4.14 p=.91</td>
<td>4.25 3.233 p=.12</td>
<td>4.18 3.48 p=.08</td>
</tr>
</tbody>
</table>

[ALL=whole group, LEC=lecture group, LAB=lab group, FD=field dep, FI=field indep,*p<.05,**p<.01]
Row one, column seven shows a statistically significant increase in the student rating for this item (p=.03*).

This response suggests that the students were angrier in the classes after the project period. This dissatisfaction may also be the result of increasing work difficulty, loss of control, and reduced progress in class. This would be consistent with the above three analyses. Other factors like the impending final examination, the submission of a major assignment and student fatigue may also have contributed to the increased feelings of anger. These will be explored in a later discussion which considers the context of the project within the overall course structure.

4.3 Lecture/Lab Contrasts

The allocation of the sixteen participants to the lecture and lab groups was based upon a programming quiz and a field independence test (Appendices E and F). The allocation also resulted in a fairly even distribution of males and females to each group (Figure 12). The lecture group contained four males and four females. The lab group contained 5 males and 3 females.

<table>
<thead>
<tr>
<th>Field Independent</th>
<th>Lecture</th>
<th>Lab</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>4 Students (2F,2M)</td>
<td>5 Students (3F,2M)</td>
</tr>
<tr>
<td>Field Dependent</td>
<td>4 Students (2F,2M)</td>
<td>3 Students (3M)</td>
</tr>
</tbody>
</table>

Figure 12. Lecture/lab: Group allocation (F=Female, M= Male).

Eight students were interviewed, four from each of the lecture and lab groups (two selected randomly from each of the four sub-groups). The statistical analyses of the attitude questionnaire and daily log sheets produced fifteen significant changes for the lecture group and two significant changes for the lab group.
4.3.1 Previous computer experience (i.e. data base, etc.)

Figure 13 shows the level of computer experience as reported by the lecture and lab groups.

![Bar Chart: Previous computer experience](image)

Figure 13. Lec/Lab: Previous computer experience.

While the level of computer experience between the two groups differed slightly, a correlation matrix to determine the relationship between experience and programming performance (the two quizzes and final exam) showed no significant correlation for the groups. Therefore previous computer experience is less likely to be a factor contributing to the differences that may occur between the students in either the lecture or lab environment.

4.3.2 Previous computer programming experience

Figure 14 shows the previous level of programming experience as captured during the student interviews. The lab group reported some formal experience with computer programming while the lecture group had either some informal or no experience at all.
Although previous student programming experience was minimal, recall that to ensure an even spread of abilities between the groups, and so control for this factor, allocation to the groups was based on the scores from the first programming quiz.

4.3.3 Self confidence when working with computers

The students were asked in the attitude questionnaire to rate their ‘self-confidence’ when working with computers. The attitude questionnaire was given on two occasions, once before the project and then again after the project. Table 9 shows a significant change for the lecture students rating of this item (p=.03*). The lab group recorded no change at all.

Table 9. Mean Student Rating Difference Between Paired Groups, Attitude, Question 38.

<table>
<thead>
<tr>
<th>Groups</th>
<th>Before</th>
<th>After</th>
<th>p</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALL (n=16)</td>
<td>3.312</td>
<td>3.562</td>
<td>.1</td>
</tr>
<tr>
<td>LEC (n=8)</td>
<td>3.125</td>
<td>3.625</td>
<td>.03*</td>
</tr>
<tr>
<td>LAB (n=8)</td>
<td>3.5</td>
<td>5.5</td>
<td>-</td>
</tr>
<tr>
<td>FD (n=7)</td>
<td>3.714</td>
<td>3.714</td>
<td>-</td>
</tr>
<tr>
<td>FI (n=9)</td>
<td>3.0</td>
<td>3.444</td>
<td>.04*</td>
</tr>
<tr>
<td>Male (n=9)</td>
<td>3.556</td>
<td>3.889</td>
<td>.08</td>
</tr>
<tr>
<td>Female (n=7)</td>
<td>3</td>
<td>3.143</td>
<td>.6</td>
</tr>
</tbody>
</table>

[ALL=whole group, LEC=lecture group, LAB=lab group, FD=field dep, FI=field indep,*p<.05,**p<.01]
Consider Table 10, a correlation matrix for the items: experience and self confidence. Column one shows a strong relationship for the lecture group both at the beginning (correlation value=.71) and at the end (correlation value=.78) of the project period. By contrast, column two shows poor relationship for the lab group.

<table>
<thead>
<tr>
<th>Items</th>
<th>Lecture</th>
<th>Lab</th>
</tr>
</thead>
<tbody>
<tr>
<td>Experience &amp; Q.38 Before</td>
<td>.71</td>
<td>.24</td>
</tr>
<tr>
<td>Experience &amp; Q.38 After</td>
<td>.78</td>
<td>.08</td>
</tr>
</tbody>
</table>

This analysis suggests that entry level experience is likely to be one factor accounting for the marked increase in the confidence level of the lecture group. The entry level experience of the lab group was higher to begin with and therefore any positive change in confidence is likely to be smaller by comparison. The resilience of the lab group to changes in their instructional environment was a characteristic noted throughout the project period.

4.3.4 Group allocation

During the interviews, the students were asked how they felt about being placed in either the lecture or lab group. Figure 15 summarizes the eight responses:

<table>
<thead>
<tr>
<th></th>
<th>Lecture</th>
<th>Lab</th>
</tr>
</thead>
<tbody>
<tr>
<td>Field Independent</td>
<td>S1: Rather the lab.</td>
<td>S5: Glad.</td>
</tr>
<tr>
<td></td>
<td>S2: Didn't mind.</td>
<td>S6: Didn't mind.</td>
</tr>
<tr>
<td>Field Dependent</td>
<td>S3: Concerned.</td>
<td>S7: Preferred this.</td>
</tr>
<tr>
<td></td>
<td>S4: Uneasy.</td>
<td>S8: Satisfied</td>
</tr>
</tbody>
</table>

Figure 15. Lec/lab: Reactions to group allocation (S=Student).
The lab participants indicated satisfaction with their allocation to the lab style learning environment. Two of the four lecture participants voiced concern over their placement:

(Lec = lecture group, Lab = lab group)

Student 8 (Lab): I was satisfied.

and,

Student 7 (Lab): I liked it a lot better than being [put] in the lecture [group].

and,

Student 5 (Lab): I was glad.

and,

Student 3 (Lec): . . . concerned about it being an advantage or disadvantage being in one group or the other.

and,

Student 4 (Lec): I'm not sure what the other people had, but uneasy.

While the concerns of the lecture group participants were initially unclear, further light was shed on this when the students were questioned on their 'preferred learning style'.

4.3.5 Preferred learning styles

Students were asked to comment on the way in which they preferred to learn things in general. The question was directed at general learning and not specifically at computer programming:

In general, given that you wanted to learn something, what learning style would you most prefer?
The students were not given a list to choose from nor directed to respond in any particular way. The question was deliberately left very open to allow for a range of possible responses. A summary of the eight student responses is recorded in Figure 16:

<table>
<thead>
<tr>
<th>Field Independence</th>
<th>Lecture</th>
<th>Lab</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>S1: To be done by doing.</td>
<td>S5: To learn as you do it.</td>
</tr>
<tr>
<td>Field Dependence</td>
<td>S3: Theory then practice.</td>
<td>S7: Theory then practice.</td>
</tr>
<tr>
<td></td>
<td>S4: Read, re-read, ask.</td>
<td>S8: Experience through involvement.</td>
</tr>
</tbody>
</table>

Figure 16. Lec/lab: Preferred learning styles (S=Student).

There was no real contrast in the way the two groups responded. What became clear is that the students indicated a preference for active student involvement in the learning environment. Clear voice was given to dissatisfaction with passive lecture-type environments. Alternative instructional environments preferred by the students included:

- 'hands-on' learning,
- seminar groups, and
- tutorial sessions.

This became even more apparent when the interviewees were asked which learning environment, either the lecture or lab, they would have chosen if given free choice.
4.3.6 Free choice:

Figure 17 summarizes the eight student responses to the interview question:

If you had free choice which [learning style, lecture or lab] would you opt for?

<table>
<thead>
<tr>
<th></th>
<th>Lecture</th>
<th>Lab</th>
</tr>
</thead>
<tbody>
<tr>
<td>Field Independent</td>
<td>S1: Lab.</td>
<td>S5: Lab.</td>
</tr>
<tr>
<td></td>
<td>S2: Lab.</td>
<td>S6: Lecture.</td>
</tr>
<tr>
<td>Field Dependent</td>
<td>S3: Lecture.</td>
<td>S7: Lab.</td>
</tr>
<tr>
<td></td>
<td>S4: Lab.</td>
<td>S8: Lab.</td>
</tr>
</tbody>
</table>

Figure 17. Lec/Lab: Given free choice, what style would you opt for? (S=Student).

The responses further highlight the student expression for active involvement in the learning situation. The students perceived this as being more likely in a lab as opposed to a lecture environment. While some indicated a preference for lectures, further questioning revealed that increased student interaction in the learning environment was clearly desirable. Student responses included:

(Student 8 (Lab): ...) the best thing is to experience [it] when I get involved, so not just when somebody tells me this is what you have to learn, learn it!

and,

(Student 7 (Lab): ...) through practice, and not so much just through lecturing, you have to be able to do something, to work with it, ... because there is more interaction between the, ah, students and the teacher.

and,
Student 4 (Lec): Its different because you get quite a bit of feedback from the computer itself in whether you are doing things correctly. So I think the way that I would prefer to learn this is just total laboratory sessions with asking questions of people as we go along.

and,

Student 6 (Lab): Because there would probably be someone there to help you.

Active student involvement and increased student-teacher interaction are clearly two factors which students see as being important in their learning environment. Both overlap with 'teacher helpfulness' and 'teaching style' as discussed earlier in this chapter in relation to the group as a whole (Figure 9).

4.3.7 Instructor's teaching style:

On the daily log sheet students were asked to record their feelings about the instructor's teaching style (Table 11). The analysis in Table 11 (column four and seven, row one) shows that students in the lecture group rated the instructor's teaching style as being better (statistically significant, p=.04*) going from the regular lectures in the project period and then worse (statistically significant, p=.04*) when moving back to regular lectures.

It would appear that the lecture group, although apprehensive about the lecture method as a teaching style, did enjoy it for the duration of the project period. Two interview excerpts give some insight into the differences perceived between the lectures held during the project period as opposed to those outside the project period:

(Lec = lecture group, Lab = lab group)

Interviewer: How did you find the notes on the board?

Student 1 (Lec): Outside the project period they were useless and frustrating, a lot of background concepts and too little practicalities. During the project period better on the practicalities.
Table 11. Mean Student Rating Difference Between Paired Groups: Log Sheet, Q.8.

<table>
<thead>
<tr>
<th>Groups</th>
<th>Before - During</th>
<th>During - After</th>
<th>Before - After</th>
</tr>
</thead>
<tbody>
<tr>
<td>All (n=16)</td>
<td>3.429  4.1   p=.007 **</td>
<td>4.054  3.062   p=.004 **</td>
<td>3.392  3.108   p=.28</td>
</tr>
<tr>
<td>LEC (n=8)</td>
<td>3.529  4.2   p=.04 *</td>
<td>4.275  3.312   p=.04 *</td>
<td>3.529  3.429   p=.81</td>
</tr>
<tr>
<td>LAB (n=8)</td>
<td>3.329  3.7   p=.11</td>
<td>3.7    2.66   p=.08</td>
<td>3.2    2.66   p=.08</td>
</tr>
<tr>
<td>FI (n=9)</td>
<td>2.914  3.971  p=.005 *</td>
<td>4.014  2.843   p=.02 *</td>
<td>2.9    2.9  p=1</td>
</tr>
<tr>
<td>Male (n=9)</td>
<td>3.556  4.289  p=.005 *</td>
<td>4.157  3.329   p=.09</td>
<td>3.529  3.329   p=.64</td>
</tr>
<tr>
<td>Female (n=7)</td>
<td>3.2    3.76  p=.34</td>
<td>3.933  2.75   p=.03 *</td>
<td>3.2    2.8   p=.11</td>
</tr>
</tbody>
</table>

[ALL=whole group, LEC=lecture group, LAB=lab group, FD=field dep, FI=field indep, *p<.05, **p<.01]

and,

Student 4 (Lec): There seemed to be a real contrast in styles where one particular lecturer made things quite complicated [outside project period] whereas in the other case the things were even perhaps put at a too simple level [during the project period].

and,

Interviewer: Do you think you asked a lot of questions?

Student 4: Depends on the lecturer.

Interviewer: When did you ask more questions then, when you were in the unnecessarily complicated or in the overly simple?

Student 4: Simple.
Student 1's reference to the regular lectures as being 'useless and frustrating' is a very strong criticism and although it must be kept in the context of the range of possible student reactions to any instructor, it does shed some light on the anti-lecture feeling that the students reported. This sentiment is also reflected in the next section which considers student expectations of a particular teaching style.

4.3.8 Was the teaching style what you expected?

In an attempt to clarify the participants' reactions to their assigned learning environment the interviewees were asked if the teaching style they experienced (i.e. in the lecture or lab) was what they had initially expected. Figure 18 summarizes the eight interviewee responses:

<table>
<thead>
<tr>
<th>Field Independent</th>
<th>Lecture</th>
<th>Lab</th>
</tr>
</thead>
<tbody>
<tr>
<td>S1: More interaction.</td>
<td>S5: As expected.</td>
<td></td>
</tr>
<tr>
<td>S2: More interaction.</td>
<td>S6: As expected.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Field Dependent</th>
<th>Lecture</th>
<th>Lab</th>
</tr>
</thead>
<tbody>
<tr>
<td>S3: As expected.</td>
<td>S7: As expected.</td>
<td></td>
</tr>
<tr>
<td>S4: More interaction.</td>
<td>S8: As expected.</td>
<td></td>
</tr>
</tbody>
</table>

Figure 18. Lec/Lab: Was the teaching style what you expected? (S=Student).

The lab participants unanimously reported that the learning environment they experienced was what they had expected. By contrast, three out of the four lecture group interviewees felt the lecture environment differed from what they initially anticipated. Their responses are given below:

(Lec = lecture group, Lab = lab group)

Interviewer: In the lecture group, was it what you expected?

Student 4 (Lec): Not a typical lecture.
Interviewer: In what ways?

Student 4: It was becoming more like a seminar.

and,

Interviewer: In the lecture group, was it what you expected it to be?

Student 2 (Lec): ... (uncertain pause) ... yes.

Interviewer: Do you want to add anything to that?

Student 2: Yes there was more interaction in your lecture session, than there have been with our normal instructor.

Interviewer: Interactions with the instructor?

Student 2: Student interactions.

and,

Student 1 (Lec): The small group lecture tended to be like a tutorial.

The lecture group participants found that their original expectations of a lecture environment differed to what they actually experienced during the project period. It should be remembered that class size was halved for the duration of the project period, one half attending lecture style instruction, the other half attending lab style instruction. Although they attended lecture sessions (Appendix A contains the lecture preparation notes used during the project period) the students reported these as being 'seminar' or 'tutorial' in style. Reduced class size may have been a factor which altered their perception of the expected lecture environment. Increased student interaction, perhaps the result of the smaller class size, may have been another factor. In earlier interview excerpts (see discussion on Whole Group analysis) the students suggested that the project lecturer developed a better rapport with the group. The following discussion on 'instructor helpfulness' provides a further insight to the favourable reaction of the lecture group to the project period lectures.
4.3.9 Instructor helpfulness:

Student response to 'instructor helpfulness' followed a similar pattern to the previous analysis (Table 5 and Figure 19). Although both groups showed a more positive response during the project, it was the drop in the lecture group rating that was statistically significant for the 'during to after' time interval (row two, column seven, p=.04*).

![Figure 19. Lec/lab: Changes in Instructor helpfulness rating.](image)

The instructor after the project was seen by the lecture group to be less helpful than the instructor during the project. This reaction parallels the earlier data indicating that the lecture group participants found the project period lectures more satisfying than the regular lectures which followed.

4.3.10 Learner control

The daily log sheets tracked student feelings about being in control of their own learning (Table 12). The lecture group rating showed a statistically significant drop as they moved from the project lectures back to the regular lectures (p=.04*).
Table 12. Mean Student Rating Difference Between Paired Groups: Log Sheet, Q.3.

<table>
<thead>
<tr>
<th>Groups</th>
<th>Before - During</th>
<th>During - After</th>
<th>Before - After</th>
</tr>
</thead>
<tbody>
<tr>
<td>All (n=16)</td>
<td>3.693 3.957 p=.14</td>
<td>3.931 3.177 p=.02 *</td>
<td>3.683 3.233 p=.07</td>
</tr>
<tr>
<td>LEC (n=8)</td>
<td>3.471 3.929 p=.06</td>
<td>3.937 3.088 p=.04 *</td>
<td>3.471 3.171 p=.43</td>
</tr>
<tr>
<td>LAB (n=8)</td>
<td>3.914 3.986 p=.79</td>
<td>3.92 3.32 p=.31</td>
<td>3.98 3.32 p=.048 *</td>
</tr>
<tr>
<td>FI (n=9)</td>
<td>3.486 3.7 p=.51</td>
<td>3.657 3.243 p=.3</td>
<td>3.403 3.367 p=.65</td>
</tr>
<tr>
<td>FD (n=7)</td>
<td>3.9 4.214 p=.09</td>
<td>4.25 3.1 p=.04 *</td>
<td>3.883 3.1 p=.07</td>
</tr>
<tr>
<td>Male (n=9)</td>
<td>3.656 3.922 p=.2</td>
<td>3.857 3.4 p=.32</td>
<td>3.629 3.4 p=.54</td>
</tr>
<tr>
<td>Female (n=7)</td>
<td>3.76 4.02 p=.49</td>
<td>4.017 2.917 p=.02 *</td>
<td>3.76 3 p=.01 *</td>
</tr>
</tbody>
</table>

[ALL=whole group, LEC=lecture group, LAB=lab group, FD=field dep, FI=field indep,*p<.05,**p<.01]

One of the interview questions which sought student reaction to 'learner control' in the instructional environment was:

During the class, would you prefer the teacher to provide the organization for you or would you rather have freedom to structure your own learning?

The eight student responses are summarized in Table 20:
Interviewee preferences ranged from 'organized' to 'freedom' with some preferring a combination of the two.

Two sample reactions indicative of the student responses were:

(Lec = lecture group, Lab = lab group)

Student 1 (Lec): I would rather have the freedom to structure my own learning. You can learn things that are more applicable and leave out the unnecessary bits.

and,

Student 4 (Lec): Freedom to structure my own learning. The things that I understand well I can move through those very quickly, with the things that I don't understand so well I can spend more time on and go at a slower pace.

These two excerpts add further support for some measure of active student participation in the instructional environment, an issue strongly felt by the majority of the interviewees earlier in the chapter (see Whole Group section).

In short, the issue of 'learner control' is not too distant from the concept of active student participation in the instructional environment. In both areas, while there was no clear difference between lecture and lab groups, the
majority opinion tended towards greater student freedom and participation in the learning environment. Both groups indicated that the regular lecture environment after the project period was more restrictive than that experienced earlier in the course.

4.3.11 Work comprehension, progress, anger

The daily log sheets also tracked three other items:

• student comprehension (Q.12),
• progress in class (Q. 7), and
• feelings of anger (Q.9).

Although the three were not found to be strongly correlated there was a positive relationship between each (Table 13). For this reason, and also because the lecture group ratings for all three showed a statistically significant decline when moving from the project period back to the regular lectures, they will be considered together. The lab group recorded no statistically significant changes for any of the items.

<table>
<thead>
<tr>
<th>Items</th>
<th>Before</th>
<th>During</th>
<th>After</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q.9 &amp; Q.7</td>
<td>.69</td>
<td>.713</td>
<td>.623</td>
</tr>
<tr>
<td>Q.9 &amp; Q.12</td>
<td>.856</td>
<td>.75</td>
<td>.416</td>
</tr>
<tr>
<td>Q.7 &amp; Q.12</td>
<td>.574</td>
<td>.631</td>
<td>.728</td>
</tr>
</tbody>
</table>

Because inability to comprehend the work is likely to result in reduced progress in class and then possibly greater anger, the analysis will consider the three issues in that order. Firstly, the ratings that the lecture group gave for their 'ability to comprehend the work' showed a statistically significant improvement (Figure 21, Table
7, p=.049*) moving into the project period and then a statistically significant decline (p=.02*) moving back to the regular lectures.

Figure 21. Lec/Lab: Changes in Ability to Comprehend the Work rating.

Secondly, Consider Figure 22.

Figure 22. Lec/Lab: Changes in 'progress in class' rating.
This result was accompanied by a significant decline in the progress the lecture group felt they made after the project period as opposed to during it (Figure 22, Table 6, p=.01*). This result is not surprising considering they reported greater difficulty in comprehending the work.

Finally, the lecture students rating of 'anger' showed a statistically significant increase as they moved from the project period back to the regular lectures (Figure 23, Table 8, p=.03*). The increase in anger parallels declining comprehension and feelings of reduced progress in the class work.

Figure 23. Lec/lab: Changes in Anger rating.

4.3.12 Daily attitude

A summary analysis of the twelve daily log sheets questions (Table 4 and Figure 24) shows similar trend to those just reported. The lecture group recorded a significant decline as they moved from the project period back to the regular lecture sessions (p=.03*).
Throughout the project period the lab group remained very resilient to their transfer from regular lectures to the lab-style instructional environment (for the project period) and then back to regular lectures. As noted earlier, the lab group was satisfied with their allocation to the lab teaching style and they found it to be as they anticipated.

4.3.13 Programming performance

The final analysis considers student performance on the course (Appendix F). The quizzes and examination tested the students ability to write, code and debug programs in the BASIC language. Quiz one was given at the beginning of the project period and quiz two at the end of the project. A final examination was given at the end of the course. Although Figure 25 shows a slight difference, there was no statistically significant difference found between the two groups.
This result suggests that students performed equally well, regardless of the teaching environment experienced (i.e. lecture or lab).

**4.3.14 Student Fatigue**

Finally, although data on student fatigue was not deliberately sought, fatigue was apparent towards the end of the course. The students involved in this project were on a tight time schedule. They had three weeks to learn and demonstrate mastery of the BASIC programming language. All eight students interviewed expressed a preference for a six or thirteen week course (Figure 5) if they were to do it again. All said that time had 'flown by' and indicated that they were under pressure. The following comments were typical of student responses:

(Student 1 (Lec): You tend to miss stuff because the work has no real chance of sinking in. Often you are not comfortable with the previous days work before having to deal with the next days work.)

Figure 25. Lec/Lab: Changes in Programming Performance.
and,

Student 2 (Lec): Too much, too fast, no absorption time.

and,

Student 5 (Lab): If you blink you'll miss it!

The data collected during the third week showed a decline in student performance and attitude. Students voiced greater dissatisfaction with several aspects of the learning environment (e.g. instructor helpfulness, ability to comprehend the work, increase in anger). This decline may have been, in part, due to fatigue resulting from the pressure of time and the intensity of the course.

4.4 Summary

The analysis of the data considered the group as a whole and then contrasted the lecture and lab sub-groups.

Students scores on the tests of programming ability ranged between 75% and 85%. There was no statistically significant difference between the scores of the lecture or lab groups.

The whole group indicated a preference for active participation in the learning environment. This was consistent with a general feeling of displeasure with passive instructional environments which allow little student control over the learning process. The majority of the group indicated that they would elect to attend a lab style instructional environment if they had been given free choice.

At the end of the project period the lab students indicated that their style of instruction was very much as they had anticipated. By contrast the lecture group found the project lectures to differ from their initial expectations. The smaller group size and increased student-teacher and student-student interactions were suggested as possible reasons for their positive reactions.
Of the materials available, the daily work sheets were reported to be the most valuable resource by both groups. The text and example programming exercises were also found to be useful to the lecture and lab groups respectively. Some students indicated that a significant omission was an answer sheet for problems assigned each day.

Chapter five will consider the analyses presented in this chapter and draw some conclusions regarding a possible instructional framework suitable for use with novice programmers.
Chapter 5

Summary, Discussion and Conclusions

The purpose of this study was to extend the knowledge of instructional methods used for novice programmers by examining the perceptions of learners to the lecture-lab and the guided self-discovery approaches. The first approach being predominantly teacher centred and the second student centred. For simplicity, the first will be referred to as the lecture method and the second as the lab method. Briefly, the lecture approach, clearly distinguished between theoretical instruction and practical application. A clear separation of classroom instruction from the 'hands-on' experience. The lab method used the 'hands-on' experience as the primary instructional strategy. Student were free to experiment, explore and discover, furnished with just enough information to ensure success.

5.1 Context of the study

A brief summary of the study, up to this point, will help put the following discussion into context. Sixteen university students volunteered for the study. They were enrolled in an introductory BASIC programming course. They were divided into two groups, one to experience the lecture method the other the lab method. To ensure an even spread of abilities between the lecture and lab groups, students were allocated based on tests of programming ability and field independence. Students attended regular classes for the first third of the course, broke into two sub-groups for the experimentation period, and returned to regular classes for the final third of the course. Two instructors were used, one for the regular classes and one for the experimentation period. The data analysis was based upon tests of programming ability, attitude questionnaires, daily log sheets and interview transcripts.

5.1.1 Constraints

Any interpretation of the results of this study must be considered in the light of the following constraints:
• the size of the group, sixteen in total,
• the size of the sub-groups, eight in each,
• two different instructors, one for regular classes, the other for the project period,
• the length of the study, one third of the total course,
• the novelty effect of the project on the participants,
• fatigue due to the intensity of the course, and
• the extra resources effect.

To counteract the possible effect that the small number of participants in the study would have on the power of the statistical analyses, data were also collected through an intensive interview programme. This two pronged approach to data collection gave strength to the final analysis of the results as the interview data supported the statistical findings.

Two instructors were used for the course. One instructor was responsible for the instruction outside the project period and the second for the instruction during the project period. The value of a single instructor for the project period meant that, although the teaching methods differed, both the lecture and lab groups had the same instructor. The student reactions to the different instructor before and after the project were monitored. It was felt that the reactions were more closely related to group size and instructional method than to the individual personality or style of the instructors.

While the length of the study, one week, was a comparatively short time span, this intensity restricted possible diffusion of the treatments between the groups. If the study had been conducted over a regular thirteen week semester, with one lecture per week, then it would have been more difficult to keep the members of the two groups from exchanging notes and talking about the two teaching styles. The intensity of the project also gave an immediacy to the reactions that might otherwise have been tempered by the passage of time in an equivalent thirteen week course.
The novelty aspect must also be considered. There was always likely to be an effect due simply to the fact that subjects knew that they are participating in an experiment and experiencing the novelty of it (otherwise known as the Hawthorne effect). The students were advised on the first day of the course about the project and were given full details and the opportunity to ask questions throughout the first week. Participation was voluntary and sixteen of the seventeen students on the course volunteered. By that time the participants had already sat one quiz and were busy with the course work. While there was still some novelty in being involved in the project, the students appeared to be less concerned about the research project and more concerned about the course work.

There was also the novelty of the treatments themselves. In an attempt to maintain normality, class and student routines were carefully organized to match those of the regular university schedules. Standard rooms and facilities were used. In sum, every attempt was made to minimize the novelty effects.

Student fatigue, particularly towards the end of the course, must also be considered as a constraint. As noted in the concluding section of chapter four, the students indicated that they were under pressure and showed a decline in daily attitude and performance which may have been caused by fatigue. This is consistent with the studies examining the effect of fatigue on student performance (Montgomery, 1983; Hattie, 1983).

Finally, the effect of extra resources must be considered. Every attempt was made to ensure that the course was identical to the regular courses offered by the university. Work sheets, handouts and programming example sheets were similar to those distributed in regular courses. If anything, the use of the text was down graded because the instructors wanted to alter the sequence and depth of the material covered. Some additional notes for the class were distributed to compensate for the reduced usage of the text. Lastly the instructors were present in the laboratory sessions more than for regular classes. This was to monitor student progress. When an instructor was approached by a student, the policy was to either refer them to the teaching assistant or to similar programming examples in the students' own notes.
5.1.2 Field independence and male/female contrasts

Although a detailed analysis of field independence and gender was not a part of this study, an analysis of test results showed that:

- there was no statistically significant difference in the test performance of field independent and field dependent students, and
- there was no statistically significant difference in the test performance of males and females.

Within the context of novice programmers, informal observation suggested that the following two areas might be worthy of future research attention:

- regardless of teaching method, field independent people preferred specific assignments while field dependent people preferred free choice, and
- regardless of teaching method, field independent people indicated higher levels of frustration than their field dependent counterparts.

5.2 Novice Programmers

The analysis of the results revealed several areas of agreement among the participants regardless of instructional method. These issues will be discussed first since they have a direct bearing on the discussion of the lecture and lab groups later in this chapter.

5.2.1 Active student participation

Borich (1986) defined direct instruction, expository or didactic teaching, as being primarily teacher centred. The teacher is the major provider of information, tightly controlling several factors in the learning environment (pacing, sequence, timing, etc.). He contrasted this with content centred learning where students are allowed to
participate in determining the form in which the learning occurred but not the substance of what is learned.

These two contrasting teaching approaches highlight the sometimes passive or active role the learner can play in the instructional environment. The students who were interviewed in this study all indicated some displeasure with the lecture environment as an instructional approach. They clearly voiced a desire for active student involvement in the learning environment, a distinct move away from teacher centred instruction towards content-centred learning. A typical example of the students' reaction (taken from the Free Choice analysis) was:

\[\text{(Lec = lecture group, lab = lab. group)}\]

\[\text{Student 8 (Lab): \ldots the best thing is to experience [it,] when I get involved. So not just when somebody tells me this is what you have to learn, learn it!}\]

An analysis of the instructional environments preferred by the students revealed three main choices: 'hands-on' laboratory sessions, seminar groups, and tutorial sessions. Students saw each of these as permitting an opportunity for active participation in, and control over, the learning environment. Dalbey and Linn (1985) reported that one way to maximize student control was to make the learner more responsible for gathering information and using feedback during the learning process. This is consistent with the results of this study. It also allowed students to move at a pace more in keeping with their rate of learning, for example (from the analysis of Learner Control):

\[\text{(Lec = lecture group, lab = lab. group)}\]

\[\text{Student 4 (Lec): The things that I understand well, I can move through those very quickly, with the things I don't understand so well I can spend more time on and go at a slower pace.}\]

While no single teaching approach is going to be suitable for all, the data suggests that the instructional environment should be arranged to permit a high degree of student participation. This does not mean that lectures should be totally discarded. An appropriate solution might be to arrange the instructional environment
so that a combination of teaching approaches be used. For example, the CSED 420 course might be best taught using a combination of lectures, labs (guided self discovery) and tutorials.

5.2.2 Greater student interaction in class

The call for greater student interaction in class was another aspect of the 'anti-lecture' sentiment voiced by the students. Students wanted the opportunity to interact with both the instructor and fellow students. In the first instance, this was indicative of the apparent lack of 'instructor helpfulness' which students perceived in formal lecture environments. They clearly felt that increased interaction with the instructor would enhance their learning.

Secondly there was a strong call for increased student-student interaction. Students wanted to share their work with each other. This is consistent with the research results reported by Lemos (1977) and reviewed earlier. For the novice, learning to program demands considerable time at a computer terminal going through a process often characterized by trial and error. Student interaction may have a positive impact on the trial and error process. Based on ten years experience in teaching computer programming and from closely monitoring the participants in this study, it is clear to me that students benefit from the sharing of the problems encountered, and solutions attempted. Students identify very quickly with the difficulties encountered by others and are often able to render valuable and direct assistance. The unique student perspective of a problem may often elude even the most erudite instructor. The exchange of ideas appears to be beneficial to all involved. This is consistent with the findings of Rogers (1981) and Marchionini (1985). Their studies showed that conferencing and discussion of class work with peers was an effective way to promote higher levels of programming efficiency among novice programmers.

In short, novice programmers, in this study, clearly favoured an instructional environment which permitted significant student-teacher and student-student interactions.
5.2.3 Work sheets

As reviewed in chapter two, Marchionini (1985) argued that success in learning to program ultimately depends on the activities students do. He suggested that class work should provide a series of increasingly complex exercises based on prior student experience. This is consistent with Ausubel's (1968) concern for attending to the experiential level of the students, also reviewed earlier. The work sheets used during the project period closely followed the developmental sequence suggested by Marchionini (1985) and the analysis showed the students rated them as the most valuable resource used on the course (Figure 11).

It was anticipated that students would successfully complete at least two to three problems within each scheduled laboratory session and that this rate of success would encourage and motivate the students during the course. The analysis showed that the work sheets appeared to achieve this aim and that the workload was reasonable. A typical response (from the analysis of Work Sheets):

(Lec = lecture group, lab = lab. group)

Student 6 (Lab): The work sheets were good, really helpful. You could do them, you could have success with them.

The analysis further showed that the students used the work sheets to monitor their daily progress in class. The value of the work sheets cannot be underestimated given the student response to them during this study. The content, style and instructional value of laboratory work sheets is an area worthy of future investigation in its own right.

5.3 The Lecture and Lab Instructional Methods

The students in the lab group were subjected to significant changes in their learning environment. In what was already a intensive programme, they were asked to spend one week in regular lectures, change over to the lab for a week and then to return to regular lectures. By contrast, the lecture group were given three weeks of straight lectures, the only change being that the middle week was taught by a different lecturer.
The analysis of the data produced some unexpected results. The most surprising was the extent to which the lecture group reacted, by comparison to the lab group, for the project period. It was anticipated that the lab group would be the ones most likely to react to their learning environment given the contrast in teaching styles they were to experience. The data analysis revealed the opposite result. It was the lecture group who recorded virtually all the significant statistical changes and were quite vocal about their learning environment. The lab group, as it turned out, were quite resilient to the transitions they experienced and recorded far fewer reactions.

5.3.1 Lecture approach

For the lecture group, the teaching style they experienced was not what they anticipated. The following excerpts from chapter four (analysis of Teaching Style Expected) recalls their reactions:

(Lec = lecture group, lab = lab. group)

Student 2 (Lec): ... there was more [student] interaction [during the project lecture session] than there have been with our normal instructor.

and,

Student 4 (Lec): [The project lectures were] not a typical lecture. it was becoming more like a seminar.

and,

Student 1 (Lec): The small group [project] lecture tended to be like a tutorial.

Although the lecture group were given formal lecture style instruction during the project period, it appeared that the group size and increased student interactions created an environment more closely akin to a seminar or tutorial environment. One result was a greater number of student interactions in class; an issue already discussed earlier and one which was viewed positively by all students. The above comments also highlight the students' preconceived idea of a lecture environment, i.e. large groups with very little student interaction.
The students also perceived the project period lectures as being more practical and simpler in presentation than the regular lectures. It would appear that these factors precipitated a positive student reaction to the project period lectures. The lecture group also felt that the work was less difficult to comprehend and that the project period instructor was more helpful.

These reactions highlight elements within the learning environment which students perceive as desirable. While the lecture approach might be 'unpopular', it has its uses. Rosenshine and Stevens (1986) and Borich (1988) suggested that the lecture method is an effective teaching style when it is used to focus on the acquisition of a limited set of predetermined facts, rules, or action sequences. If used specifically for these purposes, and incorporated with a quickly paced, highly organized set of exchanges between teacher and student, the lecture approach need not be a lengthy or boring exercise. Some elements of computer programming (e.g. syntax, language nuances etc) may be classified as either rules, predetermined facts and action sequences and therefore might be effectively taught using the lecture method. The ability to incorporate lectures into an overall instructional strategy designed to maximize student learning is the challenge for teachers of novice programmers.

5.3.2 Lab approach

The value of the lab group response was their quick acceptance and obvious satisfaction with the hands-on learning experience. An early concern of the project organizers was that the lab group might feel 'all at sea' with very little instructor support. Recall that the organizers had withdrawn all but the most basic instructional components from their environment. This meant that the lab group were left with only a set of printed programming examples and a copy of each on a computer disk (Appendix B). It was left up to the students to intuit by exploration and experimentation the programming language, concepts and usage.

At first, the students did feel the need for more instruction. This was anticipated and to some degree is the essence of the lab approach: removal of passive instruction to allow for more active student participation in the learning process. It was hoped that the students would then move to the next step, that is to explore and
experiment in an attempt to make sense of a particular programming statement or concept. Given a sufficient and reasonable time period, the students did this. By modifying and improving their first attempts they began to produce results. Perhaps their initial results were less than elegant but they were results nonetheless, based on their own exploration.

The students did learn from this process. Their isolation from the instructor did not adversely effect their learning. They were able to explore, experiment and discover. Their performance on the second programming test and final exam was equal to that of their counterparts in the lecture group (Figure.25) This underscored the power of student centred learning in this context. It is consistent with the findings of Marchionini (1985) and Dalbey et al. (1986). They argued that the concept of beginning instruction with comprehension of code would encouraged students to understand segments by executing and modifying them. Recall the following interview extract (from analysis of Work Sheets):

(Lec = lecture group, lab = lab. group)

Student 5: The example disk was a great idea.

Interviewee: In what way?

Student 5: I would, um, copy a program from it, and then work on it, change things and play around with it.

Experimentation was not always an immediately rewarding experience for the students. As Perkins et al. (1976) suggested, 'tinkering' with programs has both positive and negative effects. Critical to any program modification is the ability to track the effects of each modification. Failure to do so leads to an accumulation of 'tinkers' resulting in an entirely incomprehensible program. If teachers intend to use the lab style for sections of a course then they need to conscious of such pitfalls and render guidance to students when necessary.
In short, students indicated that the lab approach did permit active participation in and control over the learning process. Although a somewhat uncomfortable and unnerving experience at times, it enabled students to explore, to discover and to learn.

5.4 Conclusion

This study focused on novice programmers' perceptions of two instructional approaches. Data analysis was based on the reactions of the group as a whole and both lecture and lab sub-groups. The results indicate that novice programmers prefer an instructional framework based on: a teaching style which allows active student participation and substantial student-teacher interactions, a learning environment which allows student exploration and substantial student-student interactions, and resource materials which include regular work sheet (Figure 26).

Figure 26. An instructional framework for novice programmers.
Novice programmers voiced a strong preference for a teaching style oriented towards active student involvement in the learning process. Passive lecture style approaches to instruction were viewed with displeasure. For the students, a measure of active participation meant having some control over their own learning coupled with substantial interactions with the instructor during the teaching process. No single teaching method is likely to satisfy all students, nor is it likely to be appropriate for all sections of a course. Those sections which deal with rules and action sequences might be best taught using a lecture method. As students gain experience, an inquiry approach leading to discovery, where the learning context is a problem, might be more suitable (Borich, 1988). Interactions between the instructor and student are important in the learning environment. An instructor who encourages questions and permits in-class discussions is perceived as being more helpful and as having a teaching style which the students prefer.

Student learning is also dependent upon the nature of the instructional environment. Novice programmers viewed the hands-on experience as being a vital component of the learning process. Careful and deliberate planning which takes advantage of the interactive nature of the computer is essential. Laboratory sessions which leave student learning more to chance or accident need to be carefully reconsidered in the light of the importance novice programmers attach to the hands-on experience (Dalbey et al. 1986; Perkins, 1976).

The students also indicated a desire for greater student-student interaction in the learning environment. From observing the students at work, the lab setting seemed to be conducive to the sharing, discussion and conferencing of one anothers work. There was considerable interaction between the students in the lab style teaching as there was for the lecture group during their scheduled laboratory sessions. In both cases the interaction was informal and impromptu. While seminars, tutorials and lectures may be designed to allow student-student interaction, novice programmers saw the lab environment as being the preferred setting for such interaction. Teachers should consider the potential of student-student interaction and structure learning environments which at least informally, but preferably by design, allow greater student-student interaction (Rogers, 1981; and Marchionini, 1985).
Finally, novice programmers viewed the daily class work sheets as the most valuable resource available during the project period. Although initially written to provide a focus for the laboratory sessions, they became the yardstick by which students measured their daily progress and success in the subject. While teaching styles and learning environments for novice programmers have been the subject of considerable research interest, resource materials have not. The results of this study suggest that the content, nature and style of work sheets used in the laboratory sessions is worthy of further investigation.

In sum, this study examined novice programmers' perceptions of the lecture-lab (teacher centred) and guided self-discovery (student centred) teaching environments. The results indicate that while both methods have individual strengths, students prefer an instructional framework based upon active student participation in the learning process, individual student exploration, student-student and student-teacher interaction, and a system of daily work sheets to enable students to monitor their progress in the courses.
References


Mayer, R., (1976), *Some conditions of meaningful learning for computer programming: advance organizers and subject control frame order*, *Journal of Educational Psychology*, 68, pp. 143-150.


TOPIC: Conditional and Unconditional Statements

GOTO
This statement causes an unconditional transfer to the specified line number, for example:

```
220 GOTO 500
```

would cause program control to be transferred to line 500.

Excessive use of GOTO statements leads to 'spaghetti' programming. Such habits make debugging very difficult and program maintenance virtually impossible.

If possible organize your program so that the tasks that need to be done follow each other in successive order.

The use of the GOTO statement is virtually taboo, and actively discouraged, in other programming languages (e.g. Pascal).

IF GOTO
This is a conditional branching statement. If the condition is satisfied then program control will be transferred to the specified line number, otherwise the remainder of the statement is ignored and control moves to the next line.

```
120 IF A >= 5 GOTO 300
```

If A was greater than or equal to 5 then program control would be transferred to line 300. If the condition is not satisfied then the computer will proceed to the next line number (i.e. 130 in this case).

The condition may be expressed in a number of ways using the logical and relational operators, some examples include:

```
100 IF X > 2 AND Y < 5 GOTO 200
110 IF X$ = "YES" GOTO 300
120 IF MID$(A$,3,2) = "OZ" GOTO 100
130 IF INT(A/5) = A/5 GOTO 500
```

IF THEN
Similar to the above command but with the flexibility of being able to specify any legal BASIC command after the THEN statement. For example:

```
250 IF NAMES = "GO" THEN PRINT "BYE"
```

Note: There are only two versions of the IF statement - IF GOTO or IF THEN. Common errors include:

```
100 IF ANS = 2 PRINT "CORRECT"  X Incorrect!
110 IF BS = "BILL" END
```

ON GOTO
A conditional transfer statement, having the same effect as a set of IF GOTO statements. For example:
If the user inputs a 2 then control would be passed to line 600 (i.e. the second line number in the list).

This multiple IF GOTO statement is often used when creating a menu driven program:

```
100 INPUT X
120 ON X GOTO 500, 600, 700
130 ............................
140 ............................
150 ............................
500 PRINT "HI"
600 PRINT X * 30
700 END
```

**GOSUB**
An unconditional branching statement causing control to be transferred to a subroutine within the program. A subroutine is a self contained piece of code that may be called repeatedly from any part of the program. A statement at the end of the subroutine (RETURN) ensures that control is returned to the line immediately following the original GOSUB statement.

```
100 X = 5
200 Y = 16
300 GOSUB 600
400 PRINT RESULT
500 END
600 X = INT(SQR(Y)*X)
610 RESULT = X/Y
620 RETURN
RUN
1.25
```

**ON GOSUB**
This command is similar to the ON GOTO command except control is passed to a specific subroutine and then passed back when a RETURN statement is issued.

```
100 INPUT X
110 ON X GOSUB 600,650,700
120 ............................
130 ............................
140 END
600 PRINT "HI"
610 RETURN
650 PRINT "BYE"
660 RETURN
700 PRINT "SAY AGAIN"
710 RETURN
```
Decisions
Use of relational operators (see above) and also refer to pages 45 - 49 of the text examples 3.2.3, 3.2.5 and 3.2.6.

Random Numbers
RND(positive number). Whenever the argument of the random number function is a positive number (by convention we usually use a 1) it will always generate a pseudo-random between 0 and 1.

100 PRINT RND(1)
RUN .5436432

If you turn the computer off and on again, and run your program you will discover that the random numbers generated are in the same sequence each time. The following routine will ensure a totally random number is generated each time. A new random number seed is generated according to the precise time of input in line 100.

100 INPUT "ANY KEY TO CONTINUE"; B
110 G = PEEK (78) + 256 * PEEK (79)
120 K = RND(-G)
130 REM PRINT THE RANDOM NUMBER
140 PRINT RND(1)

RND(zero) always returns the last random number that was generated.

RND(negative number) always reproduces exactly the same number, therefore is not a random number generator at all. -1 produces a particular number, -2 produces a particular number etc.

INT(X)
The INT function is regularly used in combination with the RND function. INT(X) will return the integer component of X.

100 X = 12.3456
110 PRINT INT(X)
RUN 12

A random number between 0 and X
If I wanted to produce 10 random numbers between 0 and 6:

100 FOR X = 1 TO 10
110 PRINT INT (RND(1) * 7)
120 NEXT X

A random number between 1 and X
If I wanted to produce 10 random numbers between 1 and 6:

100 FOR X = 1 TO 10
110 PRINT INT (RND(1) * 6) + 1
120 NEXT X
A random number between X1 and X2
If I wanted to produce 10 random numbers between 10 and 20:

```
100    FOR X = 1 TO 10
110    PRINT INT(RND(1) * 11) + 10
120    NEXT X
```

******

89
TOPIC: Arrays (Lists), Errors

Arrays
Arrays are an extension of the simple variable. Arrays may be:

- Single dimension (list): \( A(X) \)
- Double dimension (tables): \( A(X,Y) \)
- Multi-dimensional: \( A(X,Y,Z) \) or \( A(V,X,Y,Z) \), etc.

Any array variable, e.g. \( A(X) \), is made up of two parts:

- The name: \( A \)
- The subscript: \( X \)

Both of these identify a unique variable.

An array name must follow the usual conventions for any variable name (e.g. it must start with a letter, may be real, integer or string, etc). Array subscripts must be positive whole numbers (may include 0).

Dimension Statement

If you intend to use subscripts that exceed 10, say 1 to 100 then you must dimension the array:

```
100 DIM A(100)
```

This will create 101 locations in the computer memory:

\[
\begin{align*}
A(0) &= \\
A(1) &= \\
A(2) &= \\
A(3) &= \\
A(4) &= \\
&\quad \text{and so no up to } \ldots \\
A(100) &= 
\end{align*}
\]

each of which may be assigned a separate value.

The string array \( \text{FIRSTNAME}\$(X) \) might contain the data items:

\[
\begin{align*}
\text{FIRSTNAME}\$(1) &= \text{'}MARY' \\
\text{FIRSTNAME}\$(2) &= \text{'}BILL' \\
\text{FIRSTNAME}\$(3) &= \text{'}JOHN' \\
\text{FIRSTNAME}\$(4) &= \text{'}DAVE' \\
\text{FIRSTNAME}\$(5) &= \text{'}CORY' \\
\text{FIRSTNAME}\$(6) &= \text{'}ANNE'
\end{align*}
\]

\( \text{FIRSTNAME} \) is the array name, and the numbers in the brackets are the array subscripts. The six subscripts uniquely identify each variable with its own data item.

Array Manipulation
Operation with arrays may involve such things as:
Inputting the data list directly from the keyboard:

100 FOR X = 1 TO 6
110 INPUT FIRSTNAME$(X)
120 NEXT X

Printing out the array list:

120 ......
130 PRINT "CHRISTIAN NAMES"
140 PRINT "-------------"
150 FOR X = 1 TO 6
160 PRINT X, FIRSTNAMES(X)
170 NEXT X

The output would be:

1   MARY
2   BILL
3   JOHN
4   DAVE
5   CORY
6   ANNE

You may wish to manipulate the array to search for a specific name. One way of doing this might be:

180 PRINT "WHOSE NAME DO YOU WISH"
190 INPUT "TO SEARCH FOR?"; KEY$
200 FOR X = 1 TO 6
210 IF KEYS = FIRSTNAMES(X) THEN PRINT "NAME PRESENT"
220 NEXT X
230 PRINT "SEARCH COMPLETED"
240 END

An array list may be used in a similar manner for real numbers (and integers).

The following program segment would read numbers from a data statement into the array SCORE(X), progressively sum those values, and finally output the result and the average of the numbers.

540 FOR K = 1 TO 8
550 READ SCORE(K)
560 SUM = SUM + SCORE(K)
570 NEXT K
580 PRINT "THE SUM OF ALL NUMBERS IS ";SUM
590 PRINT "THE AVERAGE IS ";SUM/8

950 DATA 23.432,1786.0,3.54,897.2
960 DATA 1.1121,5.436,6758.43,2.9834

The following program segment would check the array list N(X) and print out all elements divisible by three:
760 FOR X = 1 TO 10
770 IF INT(N(X))/3 = N(X)/3 THEN PRINT N(X)
780 NEXT X
.
.
.
.

Single dimension array lists are often used in combination, for example you may wish to enter the names and class marks of a group of students and calculate the class average.

100 DIM A(100)
110 INPUT " HOW MANY STUDENTS ?";N
120 FOR X = 1 TO N
130 INPUT NAME$(X)
140 INPUT MARK(X)
150 SUM = SUM + MARK(X)
160 NEXT X
170 PRINT
180 FOR X = 1 TO N
190 PRINT NAME$(X), MARK(X)
200 NEXT X
210 PRINT
220 PRINT "CLASS AVERAGE = ";SUM/N

Errors
See page 56 and 336 of text for a discussion on errors.

*******
TOPIC: Graphics

Low resolution (chunky blocks!)
Graphics screen 40 x 40 (X axis 0-39, Y axis 0-39). Position 0,0 is at the upper left hand corner. Position 39,39 is at the lower right hand corner. Four lines of text at the bottom of the screen.

Commands include:
GR
COLOR = n
PLOT X,Y
HLIN X1,X2 AT Y
VLIN Y1,Y2 AT X

Set low resolution graphics mode.
Set color, 0 = Black, 15 = White.
Plot a block at position X,Y.
Horizontal line from X1 to X2 at position Y.
Vertical line from Y1 to Y2 at position X.

The following program would produce a tic-tac-toe grid on the screen.

```
100  GR
110  COLOR = 15
120  HLIN 11,21 AT 14
130  HLIN 11,21 AT 18
140  VLIN 11,21 AT 14
150  VLIN 11,21 AT 18
```

Animation

Animation is created by drawing a shape, re-drawing it in exactly the same position but in black (i.e. to wipe it out) and then drawing it immediately in the next position in the original color.

To get a square block to move across the screen:

```
100  GR
110  FOR X = 0 TO 39
120  COLOR = 15
130  HPLOT X,20
140  COLOR = 0
150  HPLOT X,20
160  NEXT X
```
High Resolution Graphics

There are two possible high resolution graphics screens. HGR gives you a graphics screen with X values 0-279 and Y values 0-159 and 4 lines of text at the bottom. HGR2 gives you a complete graphics screen with X values 0-279 and Y values 0-191. For both screens 0,0 is at the upper left hand corner. Commands include:

- **HGR** Set high resolution screen 1.
- **HGR2** Set high resolution screen 2.
- **HCOLOR = n** Set color, 0 = Black, 7 = White.
- **HPLOT X,Y** Plot a dot at X,Y.
- **HPLOT X1,Y1 TO X2,Y2** Plot a line from X1,Y1 to Y1,Y2

The following program would produce a triangle:

```
100  HGR
110  HCOLOR = 7
120  HPLOT 0,0 TO 100,0 TO 50,100
     TO 0,0
```

Interesting variations include (bow-tie pattern):

```
100  HGR
110  HCOLOR = 7
120  FOR X = 0 TO 100
130  HPLOT 0, X TO 100, 100 - X
140  NEXT X
```

* * * * * * * * * *
TOPIC: Software Engineering and Modularity

Software Engineering
A set of principles for working on software projects.

"Once upon a time they made you buy the computer and then for good measure threw in a few pieces of free software. Now they get you to buy the software and as an afterthought throw in a computer!"

Structured programming
Some computer languages have very little enforced structure. BASIC has been criticized for its lack of structure (hence the term 'spaghetti' programming). Languages that encourage programmers to select and use clearly defined data and control structures are called structured programming languages. Pascal is called a structured programming language.

Unstructured languages are usually easier to "fiddle around with" in the early stages of learning to program and are therefore often used for novice programming courses. Critics claim that early exposure to an unstructured programming language is detrimental to later skill development with a structured programming language.

Top down vs. Bottom up
See notes in test book for a simple explanation - p.90.

Programmers Design Tools
Often a program is laid down in algorithmic form before being coded to a programming language. An algorithm is a step by step description of the tasks that need to be performed. The algorithmic form used should be language independent (i.e. the programmer should be able to read the algorithm and then program in any language he/she desires). There are a number of algorithmic forms in common usage, flowcharts (p.93), pseudocode (pp. 102-103) being two.

Procedures
Procedures are small self-contained programs within larger programs. Once a certain piece of code is required on two or more occasions then it is best to make it into a procedure instead of repeating the same piece of code. Procedures may be called on several times from various parts of a program to perform specific tasks (e.g. save data to a disk, print out a list, calculate a value).

GOSUB/RETURN
In BASIC procedures are known as subroutines. A subroutine is called to perform a particular task by use of the GOSUB command. The final command in a subroutine is always the RETURN command. This signals the end of the subroutine and ensures that control is passed back to the line immediately following the original GOSUB call.

The following program has one subroutine (lines 200 and 210) and it is called twice once from line 100 and then again from line 140).

<table>
<thead>
<tr>
<th>Line</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>GOSUB 200</td>
</tr>
<tr>
<td>110</td>
<td>FOR X = 1 TO 8;</td>
</tr>
<tr>
<td>120</td>
<td>PRINT &quot;+ ++++++++ +&quot;;</td>
</tr>
<tr>
<td>130</td>
<td>NEXT X</td>
</tr>
<tr>
<td>140</td>
<td>GOSUB 200</td>
</tr>
<tr>
<td>140</td>
<td>END</td>
</tr>
<tr>
<td>200</td>
<td>PRINT &quot;+ ++++++++ +++&quot;</td>
</tr>
<tr>
<td>210</td>
<td>RETURN</td>
</tr>
</tbody>
</table>
The final printout would be:

```
+ + + + + + + + +
+               +
+               +
+               +
+               +
+               +
+               +
+ + + + + + + + +
```

**ON GOSUB/RETURN**

This command operates in much the same way as does the ON GOTO statement. Please refer to Handout Sheet 8.
Appendix B. Example Sheets - Guided Self-discovery Group
Programming Examples

TOPIC: Conditional and Unconditional Statements

GOTO
Example:

<table>
<thead>
<tr>
<th>Line</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>220</td>
<td>GOTO 500</td>
</tr>
<tr>
<td>400</td>
<td>GOTO 130</td>
</tr>
</tbody>
</table>

Used within a program:

```
100  C = 1
110  PRINT C, "TERRY TOMSETT"
120  C = C + 1
130  IF C > 4 THEN END
140  GOTO 110
RUN
1    TERRY TOMSETT
2    TERRY TOMSETT
3    TERRY TOMSETT
4    TERRY TOMSETT
```

IF GOTO
Example:

<table>
<thead>
<tr>
<th>Line</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>450</td>
<td>IF X &gt; 5 GOTO 300</td>
</tr>
<tr>
<td>200</td>
<td>IF X &gt; 2 AND Y &lt; 5 GOTO 200</td>
</tr>
<tr>
<td>190</td>
<td>IF X$ = &quot;YES&quot; GOTO 300</td>
</tr>
<tr>
<td>120</td>
<td>IF MID$(A$,3,2) = &quot;OZ&quot; GOTO 100</td>
</tr>
<tr>
<td>730</td>
<td>IF INT(A/5) = A/5 GOTO 500</td>
</tr>
</tbody>
</table>

Used within a program:

```
100  C = 1
110  PRINT C, "TERRY TOMSETT"
120  C = C + 1
130  IF C > 4 GOTO 200
140  GOTO 110
200  PRINT
210  PRINT "BYE FOR NOW !! "
RUN
1    TERRY TOMSETT
2    TERRY TOMSETT
3    TERRY TOMSETT
4    TERRY TOMSETT
```

OR

```
```

BYE FOR NOW !!
```
```
100 PRINT "TYPE IN THE VALUE FOR A"  110
120 PRINT "TYPE IN THE VALUE FOR B"
130 INPUT "MY CHOICE IS ? "; A
140 PRINT
150 IF A > B GOTO 180
160 BIG = B
170 GOTO 190
180 BIG = A
190 PRINT "THE GREATER IS "; BIG
RUN
TYPE IN THE VALUE FOR A
MY CHOICE IS 14
TYPE IN THE CHOICE FOR B
MY CHOICE IS 23
THE GREATER IS 23
```

Italics indicate user input.

```
100 PRINT "These words start with
the letter 'A': "
110 C = 1
120 READ AS
130 IF LEFT$(AS,1) = "A" GOTO 150
140 GOTO 160
150 PRINT AS
160 C = C + 1
170 IF C <= 5 GOTO 120
180 DATA MAY, APPLE, FRY, ART, MEAT
RUN
These words start with the letter 'A':
APPLE
ART
```

**IF THEN**

**Example**

```
250 IF $S = "GO" THEN PRINT "BYE"
```

```
110 IF Z = 35.5 THEN READ XS
```

**Typical errors:**

```
100 IF ANS = 2 PRINT "CORRECT"
```

```
340 IF B$ = "BILL" END
```

**Used within a program:**

```
100 X = 5
110 Y = 7
120 IF X > = Y THEN PRINT "X is the one"
130 IF X < Y THEN PRINT "Y is the one"
RUN
Y IS THE ONE
```

or
100 INPUT "CONTINUE ?"; ANS
110 IF ANS = "Y" THEN PRINT "Here we
  go again." : GOTO 100
120 IF ANS < > "Y" AND ANS < > "N"
  THEN PRINT "Type a 'Y' or 'N'. " :
  GOTO 100
130 PRINT "That's all folks."
RUN
CONTINUE ? X
Type a 'Y' or 'N'.
CONTINUE ? Y
Here we go again.
CONTINUE ? N
That's all folks.

ON GOTO
Example

| 230 | ON Z GOTO 500, 550, 700, 400 |

or

| 100 | INPUT X |
| 120 | ON X GOTO 500, 600, 700 |
| 130 | ........ etc. ............. |
| 140 | ........ etc. ............. |
| 150 | ........ etc. ............. |
| 500 | PRINT "HI" |
| 510 | GOTO 100 |
| 600 | PRINT X * 30 |
| 620 | PRINT X / 30 |
| 630 | GOTO 100 |
| 700 | END |
100 PRINT
110 INPUT "YOUR NUMBER ?"; A 120
PRINT "CHOOSE FROM ...."  
130 PRINT "1. SQUARE"
140 PRINT "2. SQUARE ROOT"
150 PRINT "3. END"
160 PRINT
170 INPUT "YOUR CHOICE: "; X
180 PRINT
190 ON X GOTO 300, 400, 500
200 END
300 PRINT "SQUARE = "; A * A
310 PRINT: GOTO 120
400 PRINT "SQUARE ROOT = "; SQR(A)
410 PRINT: GOTO 120
500 PRINT "BYE FOR NOW."
510 END
RUN
YOUR NUMBER ? 81
CHOOSE FROM ....
1. SQUARE
2. SQUARE ROOT
3. END
YOUR CHOICE: 2
SQUARE ROOT = 9

CHOOSE FROM ....
1. SQUARE
2. SQUARE ROOT
3. END
YOUR CHOICE: 3
BYE FOR NOW

THE FOLLOWING SECTION (GOSUB AND ON GOSUB) MAY BE OMITTED ON THE FIRST READING OF THESE NOTES.

GOSUB / RETURN
e.g.

120 GOSUB 200
..... ..... etc ..... 
..... ..... etc ..... 
200 PRINT "HELLO THERE"
210 PRINT "THAT'S ALL . . ."
220 RETURN

FIVE7
Italics indicate user input.

101
Used within a program:

```
100  X = 5
110  Y = 16
120  GOSUB 600
130  PRINT RESULT
140  END
600  REM Subroutine to calculate result
610  X = INT(SQR(Y)*X)
620  RESULT = X/Y
630  RETURN
RUN
1.25
```

**ON GOSUB RETURN**

Example:

```
100  PRINT "Choose a number between 1 and 3."
110  INPUT X
120  ON X GOSUB 600, 650, 700
130  IF X = 3 THEN END
140  GOTO 100
600  PRINT "HI"
610  RETURN
650  PRINT "BYE"
660  RETURN
700  PRINT "SAY AGAIN"
710  RETURN
RUN
Choose a number between 1 and 3.
?1
HI
Choose a number between 1 and 3.
?3
BYE
```

Used within a program:

```
100  PRINT "WHICH OPTION ?"
110  PRINT "1. HAPPY MESSAGE"
120  PRINT "2. SAD MESSAGE"
130  INPUT "YOUR CHOICE ? "; A
140  ON A GOSUB 400, 500
150  INPUT "CONTINUE (N/Y) ? "; ANS
160  IF ANS = "N" THEN PRINT "BYE": END
170  GOTO 100
400  PRINT "SMILE"
410  RETURN
500  PRINT "FROWN"
510  RETURN
RUN
WHICH OPTION ?
1. HAPPY MESSAGE
2. SAD MESSAGE
YOUR CHOICE ? 1
SMILE
CONTINUE (N/Y) ? N
BYE
```

FIVE8

FIVE9

Italics indicate user input.
Random Numbers

RND(a positive number). Whenever the argument of the random number function is a positive number (by convention we usually use a 1, RND(1)), it will always generate a pseudo-random between 0 and 1.

```
100 PRINT RND(1)
RUN
.270011996
```

If you turn the computer off and on again, and run your program you will discover that the random numbers generated are in the same sequence each time. The following routine will ensure a totally random number is generated each time. A new random number seed is generated according to the precise time of input in line 100.

```
100 INPUT "ANY KEY TO CONTINUE"; B
110 G = PEEK (78) + 256 * PEEK (79)
120 K = RND(-G)
130 REM PRINT THE RANDOM NUMBER
140 PRINT RND(1)
RUN
.3211237
```

RND(0) always returns the last random number that was generated.

RND(a negative number) always reproduces exactly the same number, therefore is not a random number generator at all. RND(-1) produces a particular number, RND(-2) produces a particular number etc.

**INT(X)**

The INT function is regularly used in combination with the RND function. INT(X) will return the integer component of the number X.

```
100 X = 12.3456
110 PRINT INT(X)
RUN
12
```

A random number between 0 and X
For example, if you wanted to produce 10 random numbers between 0 and 6:

```
100 FOR X = 1 TO 10
110 PRINT INT (RND(1) * 7)
120 NEXT X
```

A random number between 1 and X
For example, if you wanted to produce 10 random numbers between 1 and 6:

```
100 FOR X = 1 TO 10
110 PRINT INT (RND(1) * 6 ) + 1
120 NEXT X
```

A random number between X1 and X2
For example, if you wanted to produce 5 random numbers between 10 and 20:

```
100 FOR X = 1 TO 5
110 PRINT INT(RND(1) * 11) + 10
120 NEXT X
```

**********
Programming Examples

TOPIC: Arrays (Lists), Errors

Arrays
Arrays are an extension of simple variables. Any array variable, e.g. A(X), is made up of two parts:

- The name: A
- The subscript: X

Both of these identify uniquely a single variable location in the memory of the computer.

Arrays may be:

- Single dimension (lists): A(X), which have a single subscript.
- Double dimension (tables): A(X,Y), which have two subscripts.
- Multi-dimensional: A(X,Y,Z) or A(V,X,Y,Z), etc.

An array name must follow the usual conventions for any variable name (e.g. it must start with a letter, may be real, integer or string, etc). An array subscript must be a positive whole number including 0.

Dimension Statement
If you intend to use arrays that will have subscripts exceeding 10, for example a single dimension array with subscripts 1 to 100, then you must dimension the array as follows:

```
100 DIM A(100)
```

This will create 101 locations in the computer memory:

- A(0) =
- A(1) =
- A(2) =
- A(3) =
- A(4) =
- and so on up to . . .
- A(100) =

each of which may be assigned a separate value.

For example the string array FIRSTNAMES(X) might contain the data items:

```
100 FIRSTNAMES(1) = "MARY"
110 FIRSTNAMES(2) = "BILL"
120 FIRSTNAMES(3) = "JOHN"
130 FIRSTNAMES(4) = "DAVE"
140 FIRSTNAMES(5) = "CORY"
150 FIRSTNAMES(6) = "ANNE"
```

FIRSTNAME is the array name, and the numbers in the brackets are the array subscripts. The six subscripts uniquely identify each variable with its own data item.
Array Manipulation
Operation with arrays may involve such things as inputting the data list (as given above) directly from the keyboard:

```
100 PRINT "Please enter six names:"
100 FOR X = 1 TO 6
110 INPUT FIRSTNAME$(X)
120 NEXT X
```

Printing out the array list:

```
100 PRINT "Please enter six names:"
110 FOR X = 1 TO 6
120 INPUT FIRSTNAME$(X)
130 NEXT X
140 PRINT
150 PRINT
160 PRINT "CHRISTIAN NAMES:"
170 PRINT
180 FOR X = 1 TO 6
190 PRINT X, FIRSTNAME$(X)
200 NEXT X
RUN
Please enter six names:
MARY
BILL
JOHN
DAVE
CORY
ANNE

CHRISTIAN NAMES:
1 MARY
2 BILL
3 JOHN
4 DAVE
5 CORY
6 ANNE
```

SIX1
Italics indicate user input.
You may wish to manipulate the array to search for a specific name. One way of doing this might be:

```
100 PRINT "Please enter six names:"
110 FOR X = 1 TO 6
120 INPUT FIRSTNAME$(X)
130 NEXT X
140 PRINT
150 PRINT
160 PRINT "CHRISTIAN NAMES:"
170 PRINT
180 FOR X = 1 TO 6
190 PRINT X, FIRSTNAME$(X)
200 NEXT X
210 PRINT "WHOSE NAME DO YOU WISH"
220 INPUT "TO SEARCH FOR ?"; KEY$
230 FOR X = 1 TO 6
240 IF KEY$ = FIRSTNAME$(X) THEN
250 PRINT "NAME PRESENT": END
260 NEXT X
270 PRINT "SEARCH COMPLETED, NAME NOT FOUND."
280 END
RUN
Please enter six names:
MARY
BILL
JOHN
DAVE
CORY
ANNE

CHRISTIAN NAMES:

1 MARY
2 BILL
3 JOHN
4 DAVE
5 CORY
6 ANNE
WHOSE NAME DO YOU WISH
TO SEARCH FOR ? GEORGE
SEARCH COMPLETED, NAME NOT FOUND.
```

An array list may be used in a similar manner for real numbers (and integers).
The following program segment would read numbers from a data statement into the array SCORE(X) and produce two simple statistics.

```
100 FOR K = 1 TO 8
110 READ NUM(K)
120 SUM = SUM + NUM(K)
130 NEXT K
140 PRINT "THE SUM OF ALL NUMBERS IS ", SUM
150 PRINT "THE AVERAGE IS ", SUM/8
160 DATA 23, 178, 3.5, 8.5
170 DATA 1, 5, 67, 24
RUN
THE SUM OF ALL THE NUMBERS IS 310
THE AVERAGE IS 38.75
```

The following program segment would check the array list N(X) and print out all elements divisible by three:

```
100 PRINT "PLEASE ENTER 6 NUMBERS"
110 PRINT "ONE AT A TIME, WHEN ASKED."
120 FOR X = 1 TO 6
130 INPUT "TYPE A NUMBER ?", N(X)
140 NEXT X
150 PRINT "NUMBERS DIVISIBLE BY THREE ARE:"
160 FOR X = 1 TO 6
170 IF INT(N(X)/3) = N(X)/3 THEN PRINT N(X)
180 NEXT X
RUN
PLEASE ENTER 6 NUMBERS
ONE AT A TIME, WHEN ASKED.
TYPE A NUMBER ? 2
TYPE A NUMBER ? 43
TYPE A NUMBER ? 78
TYPE A NUMBER ? 22
TYPE A NUMBER ? 90
TYPE A NUMBER ? 45
NUMBERS DIVISIBLE BY THREE ARE:
78
90
45
```
Single dimension array lists are often used in combination, for example you may wish to enter the names and class marks of a group of students and calculate the class average.

```
100  DIM A(100)
110  INPUT "HOW MANY STUDENTS ?";N
120  FOR X = 1 TO N
130  INPUT NAME$(X)
140  INPUT MARK(X)
150  SUM = SUM + MARK(X)
160  NEXT X
170  PRINT
180  PRINT "CSED-420 RESULTS"
190  FOR X = 1 TO N
200  PRINT NAME$(X), MARK(X)
210  NEXT X
220  PRINT
230  PRINT "CLASS AVERAGE = ";SUM/N
RUN
HOW MANY STUDENTS ? 4
JANE
  75
BILL
  80
SUE
  93
JAKE
  64

CSED-420 RESULTS
JANE  75
BILL  80
SUE  93
JAKE  64

CLASS AVERAGE = 78
```

**Errors**
See page 56 and 336 of text for a discussion on errors.
Programming Examples

TOPIC: Graphics

Low resolution (chunky blocks!)
Graphics screen 40 x 40 (X axis 0-39, Y axis 0-39). Position 0,0 is at the upper left hand corner. Position 39,39 is at the lower right hand corner. Four lines of text at the bottom of the screen:

Essential commands include:
GR (Set low resolution graphics mode.)
COLOR = n (Set color, 0 is Black, 15 is White.)

Graphing commands include:
PLOT X,Y  e.g.  100  PLOT 20,10
HLIN X1,X2 AT Y  e.g.  110  HLIN 5, 15 AT 20
VLIN Y1,Y2 AT X  e.g.  120  VLIN 20, 25 AT 30

The TEXT Command:
The TEXT command returns the screen display from graphics mode to normal text mode (i.e. 24 lines of text).
Animation

Animation is created by drawing a shape, re-drawing it in exactly the same position but in black (i.e. to wipe it out) and then drawing it immediately in the next position in the original color.

An example of a simple animation might be:

```
100 G R
110 FOR X = 0 TO 39
120 COLOR = 15
130 PLOT X,20
140 COLOR = 0
150 PLOT X,20
160 NEXT X
```

High Resolution Graphics (fine line drawing!)

There are two possible high resolution graphics screens. HGR gives you a graphics screen with X values 0-279 and Y values 0-159 and 4 lines of text at the bottom. HGR2 gives you a complete graphics screen with X values 0-279 and Y values 0-191. For both screens 0,0 is at the upper left hand corner.

Essential commands include:

- **HGR** (Set high resolution screen 1.)
- **HGR2** (Set high resolution screen 2.)
- **HCOLOR = n** (Set color, 0 is Black, 7 is White.)

Graphing commands include:

- **HPLOT X,Y**
  
  e.g. 100 HPLOT 100, 100

- **HPLOT X1,Y1 TO X2,Y2**
  
  e.g. 110 HPLOT 10,5 TO 20, 25
Used within a program:

```
100 H G R
110 HCOLOR = 7
120 H PLOT 50,0 TO 200,0 TO 200,120 TO 50,0
RUN
```

Interesting variations include:

```
100 H G R
110 HCOLOR = 7
120 FOR X = 0 TO 100
130 H PLOT 0, X TO 100, 100 - X
140 NEXT X
```

*******
Programming Examples.

TOPIC: Software Engineering and Modularity

Software Engineering
A set of principles for working on software projects.

"Once upon a time they made you buy the computer and then for good measure threw in a few pieces of free software. Now they get you to buy the software and as an afterthought throw in a computer!"

Structured programming
Some computer languages have very little enforced structure. BASIC has been criticized for its lack of structure (hence the term 'spaghetti' programming). Languages that encourage programmers to select and use clearly defined data and control structures are called structured programming languages. Pascal is called a structured programming language.

Unstructured languages are usually easier to "fiddle around with" in the early stages of learning to program and are therefore often used for novice programming courses. Critics claim that early exposure to an unstructured programming language is detrimental to later skill development with a structured programming language.

Top down vs. Bottom up
See notes in text book for a simple explanation - p.90.

Programmers Design Tools
Often a program is laid down in algorithmic form before being coded to a programming language. An algorithm is a step by step description of the tasks that need to be performed. The algorithmic form used should be language independent (i.e. the programmer should be able to read the algorithm and then program in any language he/she desires). There are a number of algorithmic forms in common usage, flowcharts (p.93), pseudo-code (pp. 102-103) being two. Below is an example of each and the resultant output.
Procedures (also referred to as modules or subroutines)

Procedures are small self-contained programs within the body of larger programs. If a certain piece of code is required on two or more occasions then it is best to make it into a procedure and call on it twice instead of repeating the same piece of code. Procedures may be called on several times from various parts of a program to perform specific tasks (e.g. save data to a disk, print out a list, calculate a value).

GOSUB/RETURN

In BASIC procedures are known as subroutines. A subroutine is called to perform a particular task by use of the GOSUB command. The final command in a subroutine is always the RETURN command. This signals the end of the subroutine and ensures that control is passed back to the line immediately following the original GOSUB call. (Additional sample programs are listed on sheet 5).

The following program has one subroutine, lines 200 and 210, and it is called twice once from line 100 and then again from line 140.

```
100 GOSUB 200
110 FOR X = 1 TO 8;
120 PRINT "+" +"; 
130 NEXT X
140 GOSUB 200
150 END
200 PRINT "+ + + + + + + + + +"
210 RETURN
```
The final printout would be:

```
+ + + + + + + + + +
+   +
+   +
+   +
+   +
+   +
+   +
+   +
+   +
+ + + + + + + + + +
```

**ON GOSUB RETURN**
This command operates in much the same way as does the ON GOTO statement in handout sheet 8.

(cont . . .)
100 FOR A = 1 TO 5
120 READ N$(A)
130 NEXT A
140 FOR A = 1 TO 12
150 READ S$(A)
160 NEXT A
170 DATA ALF, BEN, CHRIS, DOUG, EZRA
180 DATA 1,2,3,4,5,6,7,8,9,10,11,12
190 PRINT
200 PRINT "1. Names only"
210 PRINT "2. Stock"
220 PRINT "3. Full list"
230 PRINT "4. Exit"
240 PRINT
250 INPUT "Your choice ? "; X
260 PRINT
270 IF X < 1 OR X > 4 GOTO 100
280 IF X = 4 THEN END
290 ON X GOSUB 500, 600, 400
300 GOTO 100
400 REM *** Full list ***
410 GOSUB 400
420 GOSUB 500
430 RETURN
500 REM *** Names ***
510 FOR A = 1 TO 5: PRINT N$(A): NEXT A
520 RETURN
600 REM *** Stock ***
610 FOR A = 1 TO 12: PRINT A, S$(A): NEXT A
620 RETURN

RUN

1. Names only
2. Stock
3. Full list
4. Exit

Your choice ? 1

ALF
BEN
CHRIS
DOUG
EZRA

1. Names only
2. Stock
3. Full list
4. Exit

Your choice ? 4

******
Appendix C. Laboratory Work Sheets
WORKSHEET A

1. Write a program that will allow you to input, via the keyboard, 10 numbers and output at the end the largest number that you entered.

2. Write a program that allows you to enter two numbers. The program should then present a menu which lists four options (see below). On choosing one of the options the program should calculate and display the appropriate calculation.

   Addition
   Subtraction
   Multiplication
   Division

   Your Choice?

3. Write a program that will allow you to enter two whole numbers (X1 and X2) in ascending order and then printout ten random whole numbers within that range. The random numbers may include X1 and X2.

4. Write a program that will allow you to continuously input a series of letters until a sentinel value is entered. Your program is then to printout how many vowels and consonants were entered.

5. A grade 7 game goes like this:
   "Think of a number (say n), if it is even divide it by 2 (n = n/2), if it is odd multiply it by 3 and add 1 (n = 3n + 1). Continually repeat this process until n is reduced to 1."

   e.g. 
   n: 5
   n: 3 x 5 + 1 = 16
   n: 16 / 2 = 8
   n: 8 / 2 = 4
   n: 4 / 2 = 2
   n: 2 / 2 = 1

   Write a program that allows you to input a value for n (positive) and output each new value for n until n=1.

6. Write a program that will allow you to enter a number, say X, and then printout all the factors that number has, including 1 (i.e. check all factors for N = 1 through to X).

   (Useful statement IF INT(X/N) = X/N THEN PRINT N)

   Modify your program so that it will indicate if the number is prime or not.

*****
WORKSHEET B

1. Determine the output of the following:

```
100 DIM S(10)
110 FOR A = 1 TO 6
120 READ S(A)
130 SUM = SUM + S(A)
140 NEXT A
150 PRINT "THE SCORES ARE:"
160 FOR A = 1 TO 6
170 PRINT A, S(A)
180 NEXT A
190 PRINT "THE SUMMATION IS:"
200 PRINT SUM
210 PRINT "THE AVERAGE IS:"
220 PRINT SUM/6
230 DATA 10,5,15,25,40,5
240 END
```

2. Determine the output of the following:

```
100 FOR A = 1 TO 9
110 READ L$(A)
120 NEXT A
130 FOR X = 9 TO 1 STEP -1
140 PRINT L$(X);
150 NEXT X
160 END
170 DATA A,P,P,L,E,S,O,F,T
```

3. Write a program to read five names into the array WORD$(K) and then output the list in its original order followed by the reverse order. For example:

```
MARY
BILL
JOE
SUE
DAVE
DAVE
SUE
JOE
BILL
MARY
```

4. Write a program to read the following values, from a data list, into into the array A(X):

```
8.9, 8.9, 8.9, 9.6
```

Then sum the values (using ASUM for the progressive total) and print out the result. Immediately following this read into the array B(X) the values:

```
8.7, 9.5, 9.0, 9.1
```
Also sum these (using BSUM for the progressive total) and print out the result. Finally calculate the result of:

\[ \text{DIFF} = \text{ASUM} - \text{BSUM} \]

and print the result of DIFF. Comment on the value of DIFF.

5. Write a program to read into the array \( L(B) \) the following data items:

\[ A, B, C, D, E, F, G, H \]

and then write a program segment to manipulate \( L(B) \) to produce the following printout:

\[
\begin{array}{cccccc}
A & B & C & D & E & F \\
G & H & E & D & C & B \\
\end{array}
\]

6. Write a program which simulates a SAFEWAY docket. Read into a string array the name of the items and into a numeric array the cost of each item. Then produce the following printout.

<table>
<thead>
<tr>
<th>GRANVILLE SAFEWAY</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>BREAD</td>
<td>3.45</td>
</tr>
<tr>
<td>ORANGES</td>
<td>2.50</td>
</tr>
<tr>
<td>CHEESE</td>
<td>4.21</td>
</tr>
<tr>
<td>BISCUITS</td>
<td>3.99</td>
</tr>
</tbody>
</table>

TOTAL = $14.15

THANK-YOU FOR SHOPPING SAFEWAY

******
WORKSHEET C

1. Write a program that will randomly place 10 blocks of color on the low resolution graphics screen.

2. Determine the output of the following and suggest a more efficient program to produce the same result:

```
100  GR
110  COLOR = 15
120  FOR X = 1 TO 20
130  PLOT X,10
140  NEXT X
```

3. Determine the output of the following:

```
100  HGR
110  HCOLOR = 7
120  FOR X = 100 TO 150
130  H PLOT X-50, X TO X+50, X
140  NEXT X
```

4. Write a program in low resolution graphics that will allow the user to direct the growth of a pattern of blocks by typing "D" for down, "U" for up, "L" for left and "R" for right. Your program is to place the starting block in the center of the screen.

5. Write a program that in high resolution graphics that has a stick figure doing star jumps in the center of the screen. The sequence might look something like:

```
********
```
WORKSHEET D

1. Determine the output of the following:

   ```plaintext
   100 REM *** SUBROUTINE EXAMPLE ***
   110 READ A,B
   120 IF A > B THEN GOSUB 200
   130 IF A < B THEN GOSUB 300
   140 PRINT "THE NUMBERS ARE SEPARATED
   197 BY ",X," UNITS"
   150 END
   200 X = A - B
   210 RETURN
   300 X = B - A
   310 RETURN
   400 DATA 644,786
   ```

2. Determine the output of the following:

   ```plaintext
   100 A = 1
   110 BIG = 0
   120 READ X
   130 GOSUB 200
   140 A = A + 1
   150 IF A <= 4 GOTO 120
   160 PRINT BIG
   170 END
   200 IF X <= BIG GOTO 220
   210 BIG = X
   220 RETURN
   300 DATA 23, 78, 56, 87
   ```

3. Write a program that asks the user for a number (X < 35) and then calls up a subroutine to print a row of $'s and the value of X immediately following. For example if X was 5 the subroutine would print:

   ```plaintext
   $ $ $ $ $ 5
   ```

4. Write a program that will have three subroutines. The first is to allow the user to enter 5 names into the array NS(X). The second is to allow the user to enter a single name (ITEM). The third subroutine is to check through all the array items and to count how many times the value ITEM appears in the list. Your mainline program should inform the user about what is required of them before each subroutine is called, and it should printout any other appropriate messages.
CSED 420 (11/2): Computers for Instruction (with BASIC)

CSED 420 builds on the student's understanding of the principles of teaching. The purposes are twofold:

a) to examine the educational applications of microcomputer programs, and
b) to provide instruction in the writing of computer programs in BASIC.

The course focuses on the use of Apple Ile microcomputers and the language Applesoft BASIC. The use of other microcomputers is permitted. No prior experience with computer use is assumed. Taking this course, students will learn to:

1. use and instruct others in the use of the Apple Ile microcomputer and its peripheral devices,
2. write, modify and debug programs in Applesoft BASIC,
3. run programs written for the Apple Ile microcomputer,
4. propose alternative methods of integration or application of software into lessons, and
5. locate and select microcomputer programs suitable for use in schools.

As a part of learning to program in BASIC, students will also learn:

1. Applesoft BASIC variables, commands, statements, operators, arrays, functions, graphics,
2. 'good' programming style - appropriate variable names, top down design, flowcharting, screen management, user interaction,
3. program and file management, and
4. on-screen editing using the Global Program Line Editor (GPLE) and Applesoft Programmers Assistant (APA).

Students will become familiar with the library of educational software available in the Faculty of Education and will learn how to locate and evaluate outside information about educational software.

Texts and References

******
Appendix E. Group Embedded Figures Test
GROUP EMBEDDED FIGURES TEST (GEFT)

The GEFT was designed as an adaptation of the original individually administered Embedded Figures Test (EFT) which would make possible group testing. The test takes approximately 15 minutes to complete and is designed to measure field dependence or field independence. These two phrases represent a continuum along which an individual may be placed to characterize the extent to which his or her perceptions are dependent on (or independent) from cues in the environment.

The GEFT contains 18 complex figures. Subjects are shown a simple form and then required to identify and trace that form within the complex figure. Subjects are prevented from seeing simultaneously the simple form and the complex figure containing it.

For example:

Possible solutions:
Appendix F. Programming Quizzes
MULTIPLE CHOICE (1.5 marks each). Choose the most appropriate response:

1. Line numbers . . . .
   a) start each program statement.
   b) are not used if a program statement is a comment.
   c) dictate the order of executing program statements.
   d) none of the above
   e) both a and c.

2. Character strings . . . .
   a) are enclosed in quotation marks.
   b) can contain letters.
   c) can contain numbers.
   d) all of the above.
   e) a and b only.

3. Which of the following is an invalid numeric variable name:
   a) Z$  
   b) 6S  
   c) BE  
   d) all of the above. 
   e) both a and b.

4. REM statements:
   a) are executed by the computer.
   b) provide information to the computer.
   c) provide information to anyone reading the program.
   d) must be the first statement in a program.
   e) indicate the end of a program.

5. The arithmetic operation with the highest priority of operation is:
   a) exponentiation.
   b) addition.
   c) subtraction.
   d) multiplication.
   e) division.

6. PRINT statements:
   a) change the value of the contents in storage.
   b) erase the contents of a storage location.
   c) have no effect on the contents of storage.
   d) none of the above.
   e) either a or b.

7. The BASIC statement that allows data to be entered while a program is running is:
   a) INPUT
   b) TRACE
   c) READ
   d) DATA
   e) IF THEN
8. All of the following are relational operators except:
   a) < >
   b) =
   c) <=
   d) !=
   e) >=

9. The following program instructs the computer to count by 2.

```
100 M = 0
110 M = M + 2
120 PRINT M
130 IF M < 100 THEN 110
140 END
```

Which change will produce a program which can be used to count by A:

a) 105 READ A  
   150 DATA 3,5,8
b) 105 M = A  
   120 PRINT A
c) 105 INPUT A  
   110 M = M + A
d) 105 X = A  
   110 M = X + A

10. When the following program is run, the user enters numbers for A and B.

```
100 INPUT A, B
110 A = A + B
120 B = A - B
130 A = A - B
140 PRINT A, B
```

The computer will:
   a) Print the two numbers, the smallest first, followed by the largest.
   b) Print the two numbers the largest first, followed by the smallest.
   c) Print the two numbers in reverse order from the way they were input.
   d) Print the two input numbers in the same order as they were input.

**ERROR MESSAGES** (2 marks each). Identify the cause of the error and suggest the necessary corrections for the program to execute successfully:

11. 

```
100 X = 3
110 Y = 5
120 Z = Y / 1.34 + 5 X / (Y ^ 2)
130 PRINT "Z VALUE = "; Z  
RUN
SYNTAX ERROR IN 120
```

12. 

```
100 X$ = TIMOTHY
110 Y$ = PATTIE
120 IF X$ > Y$ THEN PRINT "WOOPS"  
RUN
TYPE MISMATCH ERROR IN 100
```
13. 

100  A = 150
120  B = 0.5 * A
130  IF A < B THEN GOTO 150
140  GOTO 110
150  END
RUN
UNDEF'D STATEMENT ERROR IN 140

14. 

100  NAMES = "SAMANTHA"
110  PRINT "HOW ARE YOU "; NAMES
120  ZENITH = 45.5
130  PRINT " YOUR ZENITH IS; ";
140  PRINT ZENITH
RUN
SYNTAX ERROR IN 110

15. 

100  A = 1
110  PRINT A, A * A
120  A + 1 = A
130  IF A > 5 THEN END
RUN
SYNTAX ERROR IN 120

PROGRAMMING QUESTIONS (5 marks each).

16. Write a program that will allow you to enter five marks and then print the sum of those marks. The screen output should be as follows:

|   12  |
|   32  |
|  14.5 |
|   65  |
|  43.2 |
THE SUM IS 166.7

17. Write a program that will read five numbers from a data statement (e.g. 150, 40, 160, 73, 200). As each is read, print the number and also the word "TWENTY" if it is multiple of 20. The screen output should look like:

150
40
TWENTY
160
TWENTY
73
200
TWENTY

18. What is the output for:

a) 

100  READ X, Y, Z
110  ANS = X + (4 * Z / (Y ^ 2))
120  PRINT ANS
130  DATA 12, 2, 3

129
b)

```
100 READ X, Y, AS
110 X = INT (X / Y)
120 PRINT MID$(AS, X, Y)
130 DATA 14, 4, ABCDEFGHIJ
```

19. A programmer wanted to write a program to produce the following output:

```
<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4</td>
<td>9</td>
<td>16</td>
<td>25</td>
</tr>
<tr>
<td>1</td>
<td>8</td>
<td>27</td>
<td>64</td>
<td>125</td>
</tr>
</tbody>
</table>
```

Unfortunately the program that was written (see below), although containing no syntax errors, did not produce the desired output because the lines were in the wrong order. Consider the program and suggest the necessary alterations that would produce the desired output.

```
100 A = 1
120 PRINT "TABLE OF SQUARES AND CUBES"
130 A = A + 1
140 IF A > 5 GOTO 100
150 END
```

GENERAL QUESTIONS (2.5 marks each).

20. Explain exactly what is meant by the phrase '16K of RAM'.

21. Name four restraints that need to be kept in mind when selecting a variable name.

22. Explain, briefly, what occurs during the initialization of a disk.

23. What are the three legal data types, and how are the corresponding variable names identified, in Applesoft BASIC.

******

130
University of British Columbia
CSED-420 Computers for Instruction
Brief Answers to Programming Quiz Number One


11. Missing * / 5*X
12. Missing " / "TIMOTHY"
13. Missing line 110 / 110 REM
14. Should be PRINT not PRINT
15. Should be A = A + 1

16. 100 FOR X = 1 TO 5
110 INPUT A
120 SUM = SUM + A
130 NEXT X
140 PRINT "THE SUM IS "; SUM

17. 100 FOR X = 1 TO 5
110 READ A
120 PRINT A
130 IF INT (A/20) = A/20 THEN PRINT "TWENTY"
140 NEXT X
150 DATA . . . . . . . etc . . . . . .

18.a) 15  
18 b) CDEF

19. 100 A = 1
110 PRINT " . . . heading . . . "
120 PRINT A, A*A, A*A*A
130 A = A + 1
140 IF A < 5 GOTO 120
150 END

20. 16 x 1024 bytes of random access memory (available to user).
21. Begins with a letter, no reserved words, max 238 characters, first two letters significant, no special characters (except $ and %), alphanumeric characters only.
22. Concentric circles (magnetic surface) for recording, Disk operating System laid down, Directory listing track.
23. A$ = string, A% = integer, A = real.

*******
MULTIPLE CHOICE (1.5 mark each). Choose the most appropriate response:

1. Data statements . . . .
   a) are used with INPUT statements.
   b) contain only numeric constants.
   c) dictate the order of executing program statements.
   d) are used with READ statements.
   e) both a and d.

2. The TAB function . . . .
   a) is enclosed in quotation marks.
   b) is used only with commas.
   c) causes a carriage return.
   d) enables a printed output position to be specified.
   e) a and b only.

3. Which of the following permits conditional transfer:
   a) IF X = 5 GOTO
   b) GOTO 250
   c) END
   d) PRINT " EXECUTE LINE 100"
   e) GOSUB 1000

4. REM statements:
   a) are executed by the computer.
   b) provide information essential to the computer.
   c) provide information to anyone reading the program.
   d) must be the first statement in a program.
   e) indicate the end of a program.

5. A piece of data inserted to indicate the end of a data list is called:
   a) exponentiation.
   b) progressive summation.
   c) a sentinel value.
   d) a boundary loop.
   e) an unconditional transfer.

6. Arrays are distinguished by:
   a) simple variable names.
   b) a variable name and a subscript.
   c) a FOR/NEXT loop.
   d) an argument enclosed in brackets.
   e) a variable dollar sign.

7. The step size in a FOR statement . . . . . . . . .:
   a) may be fractional.
   b) may be positive.
   c) may be negative.
   d) all of the above.
   e) both b and c.
8. All of the following are operators except:
   a) NOT  
   b) $ 
   c) <= 
   d) AND  
   e) + 

9. The following program instructs the computer to print out multiples of seven:

   ```
   100 PRINT "ENTER WHOLE NUMBERS"
   110 INPUT M
   120 PRINT
   130 IF INT(M/7) = M/7 THEN PRINT M
   140 GOTO 100
   ```

Which change will produce a program which can print out odd numbers:

   a) 110 M = 1
       120 PRINT "THE ODD NUMBERS ARE: " M
       130 M = M + 1

   b) 110 INPUT "YOUR NUMBER"; M
       130 IF M/2 <> INT(M/2) THEN PRINT M

   c) 110 INPUT A
       120 M = M + A
       130 PRINT "ODD NUMBER = " M

   d) 110 INPUT M
       130 IF INT(M/2) = M/2 THEN PRINT M

10. When the following program is run, the user enters numbers for A and B.

   ```
   100 INPUT A, B
   110 T = A
   120 A = B
   130 B = T
   140 PRINT A,B
   ```

The computer will:

   a) Print the two numbers, the smallest first, followed by the largest.
   b) Print the two numbers, the largest first, followed by the smallest.
   c) Print the two numbers in reverse order from the way they were input.
   d) Print the two input numbers in the same order as they were input.
ERROR MESSAGES (2 marks each). Identify the cause of the error and suggest the corrections necessary to overcome the error:

11.

```plaintext
100 FOR X = 1 TO 15
110 READ A$(X)
120 NEXT X
130 DATA A, B, C, D, E, F, G, H, I, J, L, M, N, O, P
RUN
BAD SUBSCRIPT ERROR IN 110
```

12.

```plaintext
100 FOR A = 1 TO 4
110 X = A
120 FOR B = 1 TO 3
130 SUM = SUM + X
140 NEXT A
150 NEXT B
RUN
NEXT WITHOUT FOR ERROR IN 150
```

13.

```plaintext
100 GR
110 COLOR = 15
120 FOR X = 40 TO 1 STEP -1
130 PLOT X, X
140 NEXT X
RUN
ILLEGAL QUANTITY ERROR IN 130
```

14.

```plaintext
100 A = 2
110 DIM NAME(A$)
120 FOR Z = 1 TO A
130 INPUT NAMES(Z)
140 NEXT Z
RUN
TYPE MISMATCH ERROR IN 110
```

15.

```plaintext
100 A = 4
110 B = 2
120 IF A > B PRINT "A LARGER": GOTO 140
130 PRINT "B LARGER"
140 END
RUN
SYNTAX ERROR IN 120
```
PROGRAMMING QUESTIONS (5 marks each).

16. Write a program in high-resolution (HGR) graphics that will draw a large diamond in the center of the screen.

17. Write a program that will read 10 people's first names into the array N$(X)$, and then write a subroutine that checks the array to see how many first names end with a vowel. Output appropriate messages.

18. Write a program to output the first 100 terms of Fibonacci's sequence (where each successive term is the sum of the preceding two terms): $1, 1, 2, 3, 5, 8, 13, \ldots$.

19. Determine the output for the following program:

```
100 S = 3
110 R = 1
120 FOR C = 1 TO R
130 PRINT S; " ";
140 S = S - 1
150 NEXT C
160 PRINT
170 R = R + 1
180 IF R <= 2 GOTO 120
```

20. Write a program that will generate 6 random numbers between 100 and 120 inclusively. The numbers may not be repeated.

21. In general, what does the following program do:

```
100 INPUT "Enter a positive 3 digit number"; X
110 H = INT( X /100)
120 T = INT ((X - 100 * H)/10)
130 U = X - (H * 100 + T *10)
140 PRINT H; "-" ; T ; "-"; U
```

22. Write a subroutine (not a full program) that will sort the array ITEM$(X)$ into ascending order. Your subroutine should begin at line 1000.

*****
University of British Columbia
CSED-420 Computers for Instruction
Brief Answers to Programming Quiz Two

1. D
2. D
3. A
4. C
5. B
6. C
7. A
8. B
9. C
10. B

11. 50 DIMA$(15)
12. 140 NEXT B
150 NEXT A
13. FOR X = 39 TO 0 STEP -1
14. 110 DIM NAME$(A)
15. IF A > B THEN PRINT "A . . . . . . etc"
16. 100 HGR
110 HCOLOR = 7
120 HPLOT 140,30 TO 180,70 TO 140,110 TO 100,70 TO 140,30
17. 100 FOR X = 1 TO 10
110 READ N$(X)
120 NEXT X
130 GOSUB 200
140 DATA TEX,SU,JA,JOY,DI,AL,KO,RAY,FI,CON
150 END
200 FOR X = 1 TO 10
210 C$ = RIGHT$(N$(X),1)
220 IF C$="A" OR C$="E" OR C$="I" OR C$="O" OR C$="U" THEN PRINT N$(X)
230 NEXT X
240 RETURN
18. 100 COUNT = 2
110 T1 = 1
120 T2 = 1
130 PRINT T1 : PRINT T2
140 FOR X = 1 TO 98
150 NTERM = T1 + T2
160 T1 = T2
170 T2 = NTERM
180 PRINT NTERM
190 NEXT X
19. 3
2 1
20. 100 FOR X = 1 TO 6
110 N = INT(RND(1)*21)+100
120 FOR Y = 1 TO X
130 IF R(Y) = N GOTO 110
140 NEXT Y
150 R(X) = N : PRINT R(X)
160 NEXT X

136
21. Places a "," between the digits of a three digit number, e.g. 5-3-8

22.  1000  F = 0  
    1010  FOR X = 1 TO N-1  
    1020  IF A(X)>A(X+1) THEN T=A(X): A(X)=A(X+1): A(X+1)=T: F=1  
    1030  NEXT X  
    1040  IF F = 1 GOTO 1000  
    1050  RETURN

*******
Multiple Choice Segment - 20 questions, 1 mark each.

1. What will be in the storage locations X, Y, and Z after the following program is run?
   10 READ U, V, W, X, Y, Z
   20 PRINT U, V, W
   30 PRINT X, Y, Z
   40 RESTORE
   50 READ V, Y, W
   60 PRINT X, Y, Z
   70 DATA 10, 20, 30, 40, 50
   80 DATA 60, 70, 80, 90, 100
   90 END
   a. 40, 50, 60
   b. 20, 30, 40
   c. 40, 20, 60
   d. 20, 40, 90
   e. 10, 30, 60

2. What will be the value of K1 after the following program is executed?
   10 LET C = 0
   20 IF C = 6 THEN 99
   30 READ K1, K2, K3
   40 PRINT K1, K2, K3
   50 LET C = C + 2
   60 GOTO 20
   70 DATA 1, 2, 3
   80 DATA 4, 5, 6
   90 DATA 7, 8, 9
   99 END
   a. 1
   b. 4
   c. 7
   d. 0
   e. none of the above

3. The statement 10 DIM A(5,5) sets aside how many locations?
   a. 5
   b. 10
   c. 25
   d. 30
   e. 36

4. The array JS$ (15) can hold
   a. fifteen numeric values or less
   b. fifteen string values or less
   c. only fifteen string values
   d. only string values
   e. none of the above
5. What will be the value of K1 after the following program is executed?

10 LET C = 4
20 IF C = 6 THEN 90
30 READ K1, K2
40 LET C = C + 1
50 GOTO 20
60 DATA 1, 2, 3, 4, 5, 6, 7, 8, 9
70 RESTORE
80 READ K1, K2
90 PRINT K1
99 END

a. 1
b. 3
c. 5
d. 4
e. 0

6. The low resolution screen is divided into _____ columns and _____ rows.

a. 48, 40
b. 40, 48
c. 24, 40
d. 40, 40
e. 40, 24

7. What will happen when the GR command is executed?

a. The computer will stop and wait for instructions.
b. The screen will go blank except for the four lines of text window.
c. The computer will enter high resolution graphics.
d. nothing
e. Any graphics program in RAM will be executed.

8. The result of the operation \(1 + 24 / 2 \times 3 \times 3\) is

a. 2
b. 12
c. 10
d. 6
e. none of the above

9. Assume that X contains 9.8, \(X = 2 \times \text{INT}(X)\) will change X to

a. 19.6
b. 19
c. 18
d. 20
e. 9

10. What does the following program print?

110 FOR I = 1 TO 2
120 FOR J = 1 TO 3
130 PRINT "X ";
140 NEXT J
150 NEXT I
160 END
11. INPUT G will
   a. put G into the RAM memory of the computer
   b. read a number from a DATA statement
   c. put the next number typed (followed by a "return") into G
   d. print G
   e. make G = 0

12. The statement R = INT(RND(1) * 5) will generate
    a. a random number from 0 to 4.99999
    b. a random number from 0 to 5
    c. a random integer from 3 to 5
    d. a random integer from 1 to 5
    e. a random integer from 0 to 4

13. The algebraic expression \( \frac{a}{b^c} \) written in BASIC is
    a. A/BC
    b. A / B * C
    c. A / B * A / C
    d. A / (B * C)
    e. A * B / C

14. DATA items cannot be
    a. letters
    b. numbers
    c. integers
    d. strings
    e. all of the above are allowed

15. The BASIC statement 10 A$ = A will cause which of the following error messages?
    a. BAD SUBSCRIPT ERROR IN 10
    b. ILLEGAL QUANTITY ERROR IN 10
    c. TYPE MISMATCH ERROR IN 10
    d. UNDEFINED STATEMENT ERROR IN 10
    e. SYNTAX ERROR IN 10

16. If the following program segment is executed:
    10 L = 0
    20 L = L + 2
    30 M = L + 3
    40 IF M <> 11 THEN GOTO 20
    50 PRINT L, M
the values of L and M that will be printed respectively are:

a. 8, 11  
b. 11, 8  
c. 14, 11  
d. 11, 14  
e. none of the above

17. After execution of the following:
   10 X = 1  
   20 FOR I = 1 TO 3  
   30 X = X * X  
   40 NEXT I  
   50 PRINT I  
   60 END

   the value of I is
   a. 1  
   b. 2  
   c. 6  
   d. 3  
   e. 4

18. To put the square root of M into K write
   a. K = "M  
   b. K = M^2  
   c. K = SQR(M)  
   d. SQRT(M) = K  
   e. K = SQR(M)

19. Which of the following is not true of sequential files?
   a. To recall an item on a sequential file, you must start at the beginning of the file and read all records preceding the desired item.
   b. Data items stored sequentially on disk are recorded one after another on the tracks of the disk.
   c. Data can be stored sequentially on both magnetic tape and disks.
   d. All sequentially stored data items may be read without reading any of the other data items.
   e. All of the above are true.

20. Closing a sequential file:
   a. prevents the loss of its contents.
   b. allows data to be written to the file.
   c. indicates to the computer that the use of the file is finished.
   d. both a and b.
   e. none of the above.

Short Answer Segment - Four questions, 5 marks each.

1. For each of the following, indicate the error in the program segment and make the correction necessary.

   A. 10 FOR I = 1 TO 5  
      20 FOR J = I TO 5  
      30 PRINT " THE SUM OF I AND J IS "; I + J  
      40 NEXT I  
      50 NEXT J
B. 10 FOR X = 1 TO 10
20 :::: PRINT X;
30 NEXT X
40 GOTO 120
50 END

C. 10 DIM X (300)
20 FOR I = 1 TO 500
30 :::: READ X (I)
40 NEXT I
50 END

D. 10 REM ***SUM THE ELEMENTS OF ARRAY***
20 DIM X(10)
30 T = 0
40 FOR I = 1 TO 10
50 T = T + X
60 NEXT I
70 PRINT "THE SUM IS "; T

E 10 DIM A(5,4)
20 FOR I = 1 TO 4
30 FOR J = 1 TO 5
40 READ A(I,J)
50 NEXT J
60 NEXT I

2. Distinguish between the following pairs of terms:
   a) NEW, HOME
   b) CATALOG, LIST
   c) CTL-C, CTL-RESET
   d) VARIABLE, VALUE
   e) GR, HGR

3. Briefly describe two methods of software evaluation, and comment on the advantages and disadvantages of each.

4. Saving text to a disk requires special DOS commands to be used within a BASIC program. Describe the commands and illustrate their use with a small program segment.

Programming segment - 60 Five questions, 12 marks each.

Do question 1, and four (4) others. All questions are of equal value. Only question 1 and four others will be marked. Part marks may be awarded for part answers.

All are to do Question One.

1. Write a program in BASIC that accepts a letter and a number as input, and produces a reverse triangle starting with the input letter and proceeding through the successive letters of the alphabet as designated by the number. For example, if A and 5 were the input,
A
BA
CBA
DCBA
EDCBA

would be the output.

Do four (4) of the remaining questions.

2. Write a subroutine that asks the user to select a letter. The subroutine then informs the user which letter was selected and asks if the user wants to change the selection. If the answer is YES, the program again asks the user to select a letter. If the answer is NO, control is returned to the main program. If the answer is neither YES nor NO, the user should be questioned appropriately.

3. Write a program which asks the student to type their two names in reverse order, separated by a comma, e.g. SMITH, SALLY. Your program will then print a welcome with the student's name in natural order surrounded by a box of asterisks which fits the name exactly.

   e.g.

   ************************
   *WELCOME SALLY SMITH*
   ************************

   The printing must be centred on a 40 column screen of 24 rows which contains no other writing.

4. Write a program which will sort the numbers contained in an array A(N), so that the even numbers are in the first part of the array in ascending order, and the odds are in the last part of the array in descending order. No other array may be used, e.g. 7, 2, 4, 9, 3, 8, 5 would become 2, 4, 8, 9, 7, 5, 3

5. Write a program for a 10 by 10 array, which prints out X's in a cross shaped pattern with intersection at coordinates given by the user, and 0's elsewhere, e.g. if the user input 4, 2 the output would be

   0 x 0 0 0 0 0 0 0 0
   0 x 0 0 0 0 0 0 0 0
   0 x 0 0 0 0 0 0 0 0
   x x x x x x x x x
   0 x 0 0 0 0 0 0 0 0
   0 x 0 0 0 0 0 0 0 0
   0 x 0 0 0 0 0 0 0 0
   0 x 0 0 0 0 0 0 0 0
   0 x 0 0 0 0 0 0 0 0
   0 x 0 0 0 0 0 0 0 0

6. 5! (read as 5 factorial) is 5 x 4 x 3 x 2 x 1 = 120. Write a program to find X factorial, where X is input from the user. The output should indicate the input value X and the value of the factorial.

7. Write a program in high resolution graphics that will print 10 (ten) lines on the screen. The start and end points of each line is to be randomly selected by the program.

   ************
Appendix G. Student Attitudes Towards Computing.
SURVEY OF ATTITUDES TOWARD LEARNING ABOUT
AND WORKING WITH COMPUTERS
FRONT PAGE FOR INITIAL QUESTIONNAIRE

The purpose of this survey is to gather information concerning people's attitudes toward learning about
and working with computers. It should take about five to ten minutes to complete. All responses will
be confidential. Please check the blanks which apply to you:

1. Age: ☐ 22 or less ☐ 23-25 ☐ 26-30
   ☐ 31-35 ☐ 36-40 ☐ 41-45
   ☐ 46-50 ☐ 51-55 ☐ 55 +

2. College level completed:
   ☐ 1st year ☐ 2nd year ☐ 3rd year
   ☐ 4th year ☐ 5th year ☐ Bachelors
   ☐ Masters ☐ Doctorate

3. Major Area of study:

   [Blank Line]

4. Gender: ☐ Female ☐ Male

5. Experience with learning about or working with computers:
   ☐ 1 week or less ☐ 1 week to 1 month
   ☐ 1 month to 6 months ☐ 6 months to one year
   1 year or more

6. Briefly state the type of computer experience:

   [Blank Line]

7. Student identification number: 145
SURVEY OF ATTITUDES TOWARD LEARNING ABOUT
AND WORKING WITH COMPUTERS
INITIAL AND FINAL QUESTIONNAIRE

Below is a series of statements. There are no correct answers. They are designed to permit you to
indicate the extent to which you agree or disagree with the ideas expressed. Please circle the label
which is closest to your agreement with each of the statements.

SA= Strongly Agree, A=Agree, U=Undecided, D=Disagree, SD=Strongly Disagree

1. Computers do not scare me at all. SA A U D SD
2. I am no good with computers. SA A U D SD
3. I would like to work with computers. SA A U D SD
4. I will use computers many ways in my life. SA A U D SD
5. Working with a computer would make me very nervous. SA A U D SD
6. Generally I would feel OK about trying out a new problem on the computer. SA A U D SD
7. The challenge of solving problems with computers does not appeal to me. SA A U D SD
8. Learning about computers is a waste of time. SA A U D SD
9. I do not feel threatened when others talk about computers. SA A U D SD
10. I don't think I would do advanced computer work. SA A U D SD
11. I think working with computers would be enjoyable and stimulating. SA A U D SD
12. Learning about computers is worthwhile. SA A U D SD
13. I feel aggressive and hostile towards computers. SA A U D SD
14. I am sure I could do work with computers. SA A U D SD
15. Figuring out computer problems does not appeal to me. SA A U D SD
16. I'll need a firm mastery of computers for my future work. SA A U D SD
17. It wouldn't bother me at all to take computer courses. SA A U D SD
18. I'm not the type to do well with computers. SA A U D SD
19. When there is a problem with a computer that I can't immediately solve, I would stick with it until I have the answer. 

20. I expect to have little use for computers in my daily life. 

21. Computers make me feel uncomfortable. 

22. I am sure I could learn a computer language. 

23. I don't understand how some people can spend so much time working with computers and seem to enjoy it. 

24. I can't think of any way that I will use computers in my career. 

25. I would feel at ease in a computer class. 

26. I think using a computer would be very hard for me. 

27. Once I start to work with the computer, I would find it very hard to stop. 

28. Knowing how to work with computers will increase my job possibilities. 

29. I get a sinking feeling when I think of trying to use a computer. 

30. I could get good grades in computer courses. 

31. I will do as little work with computers as possible. 

32. Anything that a computer can be used for, I can do just as well some other way. 

33. I would feel comfortable working with a computer. 

34. I do not think I could handle a computer course. 

35. If a problem is left unsolved in a computer class, I would continue to think about it afterward. 

36. It is important to me to do well in computer classes. 

37. Computers make me feel uneasy and confused.
38. I have a lot of self-confidence when it comes to working with computers.

39. I do not enjoy talking to others about computing

40. Working with computers will not be important to me in my life's work.

Thank-you for completing this form. Please place it in the envelope provided, seal it and return it to the instructor.
Appendix H. Daily Student Log
Daily Student Log

Date: _____________________________

Lesson Number: ____________________________

Student I.D. Number: ___________________________

1. Please estimate the time you spent on CSED-420 since the last lab session:
   Working with a computer: ________ hours.
   Working without a computer: ________ hours.

2. Please indicate, in point form, the four most important things you learned during today's class and lab session,
   e.g. a. the use of NEW,
        b. construction of a FOR/NEXT/STEP loop,
        c. a bubble sort routine.
        d. what a subscript is.

   _____________________________
   _____________________________
   _____________________________
   _____________________________

3. In point form, please list the main things which you felt the instructor was trying to teach but failed to effectively do so:

   _____________________________
   _____________________________
   _____________________________
   _____________________________

   _____________________________
   _____________________________
   _____________________________
   _____________________________
4. Please indicate how you felt about today's class:

SA = Strongly Agree, A = Agree, U = Undecided, D = Disagree, SD = Strongly Disagree

1. I felt constantly frustrated. 
2. I felt I learnt a lot.
3. I felt I was in control of my own learning.
4. I felt the instructor was helpful.
5. I felt good about my work.
6. I felt enthusiastic.
7. I felt I made very little progress.
8. I was happy with the instructor's teaching style.
9. I felt angry.
10. I was left to work on my own too much.
11. I felt I really didn't know what I was doing.
12. I felt the work was difficult to comprehend.

5. Do you have any other comments about:

- the teaching method itself,
- the learning situation,
- your own feelings about how you go about learning,
- any other observation:

__________________________________________________________________________
__________________________________________________________________________
__________________________________________________________________________
__________________________________________________________________________
__________________________________________________________________________

Thank-you for completing this form. Please place it in the envelope provided, seal it and return it to the instructor.
Appendix I. Interview Schedule
Date: __________________

Name of interviewee: __________________

Group Designation: __________

Background information:

1. Could you briefly tell me why you taking the CSED-420 course?
   - Is it part of a program?
   - Which program?
   - What year?
   - Is it for interest only?
   - Is it a compulsory or elective course?

2. If you are working, could you please describe the work you presently do?
   - Where?
   - For how long?
   - Do you enjoy it?
   - Are you considering any career change?

3. Could you please describe your teaching experience to date?
   - None?
   - Practicum only?
   - Elementary or secondary?
   - College?
   - Adults?

4. What are you subject concentrations?
5. Could you describe your present involvement in computers.
   - Do you intend to move into?
   - Computer studies education?
   - Across curricula usage?
   - School administration?

6. Do you see yourself as being increasingly involved with computers?
   - In what way?

7. Please describe your present access to computer(s).
   - At home?
   - At work?
   - Do you own one, or intend to buy one in the near future?

8. Could you please describe your previous programming experience?
   - What type?
   - What language?
   - Has it helped you much in this course?

9. Have you done summer (or inter-session) courses before?
   - How many?
   - How do you find them?

10. How do you feel about having to do CSED-420 in three weeks?
    - If you had a preference how would you like to take this course (3 weeks, 6 weeks, 13 weeks)?
Approaches to Learning:

11. In general, how would you describe the way you approach things in life?
   - radical?
   - traditional?
   - conservative?
   - experimental?
   - a 'romantic'?
   - competitive?

12. Could you briefly describe the sort of person you think is most suited to computers?
   - Do you consider yourself to be a 'computer-type' person?
   - Do you think that the lecturer and course T.A. are 'computer-type' people?

13. Do you consider yourself a 'sciences' or 'humanities' type person?
   - By watching you work, how could I tell that you are this type of person?

14. For you, what is the most efficient way to learn things?
   - Do you think this method of learning is reflected in your own teaching style.

15. If someone said you were a high achiever, how would you respond?

16. Are you the sort of person who does assignments ahead of time or do you tend to leave them to the last minute?
   - Why is this?
17. When approaching a new topic, do you prefer to be given defined goals or would you rather experiment with the topic yourself first?
   - Why?
   - What is the difference for you?

18. When given set assignment work, would you rather a specific assignment or to be given free choice?
   - Why?
   - What mode, for setting assignments, do you think you would use if you were teaching this course?

19. Do you like being given a letter grade or a numerical score on tests and assignments?
   - Why?
   - What difference is there, for you?

20. Do you consider yourself a factual/analytical type of person or one given to the approaching topics from a broader and more theoretical framework?
   - Which do you feel is the better way for you to learn?
   - Would you prefer this approach in all courses?

21. During the class, would you prefer the teacher to provide the organization for you or would you rather have freedom to structure you own learning?
   - Why?
   - What do you see are the benefits?

22. Are you the sort of person who seeks help or did you tend to 'nut' things out for yourself?
   - Where do you seek help?
23. What environment best suits your learning? How do you feel about:
   - lectures?
   - lab?
   - tutorials?
   - seminars?

24. Do you prefer quite environments for learning or do you, for instance, do you like listen to music while you study?
   - Do you find yourself easily distracted or can you cut yourself off completely from what is going on around you?

25. In this course, would you prefer to work on your own (1 to 1 with a computer) or be a partner in a group of two (2 to 1 on a computer)?
   - Why?
   - Did you seek much help from other class members?
   - Did you receive much help from other class members?
   - What sort of help?

26. In general, did you find other people around you a help, neutral or a hindrance to your learning?
   - In what way?

27. Did you think that there was much interaction between the members of the CSED-420 group?
   - Was the interaction very helpful to you?
   - Do you think the class should have been organized to allow greater interaction?
The project:

28. How did you feel about being asked to participate in the project?

29. Do you recall the Group Embedded Figures Test? Out of a possible score of eighteen, how many do you think you drew correctly?

30. You were put in the Lab or Lecture Style group. Could you describe to me your feelings about being put in that group?
   - If you had free choice which would you opt for?
   - To what extent was the style used, what you expected?

31. At any stage did you feel like opting out of the project?
   - Why?

32. Could you describe how you felt about the work load expected of you in this course?
   - Has the course been a lot of work, the same, or less than other three week courses you have taken?
   - Do you think the work load been reasonable?
   - In what ways would you change the work load?
   - Do you think you spent an excessive amount of time on this subject?
   - Did you find time 'drag' or 'fly by' for you?

33. What did you think about the course materials, handouts and references?
34. If you could comment on them individually. Specifically:
   - the textbook
   - the handout sheets
   - the work sheets
   - the student information booklet
   - the example problem sheets (G. S-d only)
   - the software disk (G. S-d only)
   - notes on the board (L-l only)

35. So, of all the materials which did you find the most useful?
   - Why?
   - Would you have preferred more, the same or less?

36. Did you use other resources beyond these?
   What were they?

37. How did you find the programming quizzes?

38. Could you please comment on whether or not you felt the instructional component of the lessons as presented made things unnecessarily complicated or perhaps overly simple?
   - Do you think you asked a lot of questions?
   - Did you have sufficient information to make some headway once the lab started or did you feel 'all at-sea'?
   - How did you feel about the content for each lesson, was it too much, just right or too little?
39. Do you feel you have been successful in this course so far?
   - How do you judge this?
   - Have you achieved more or less than you anticipated?
   - How do you feel about that?

40. During the lessons (as part of the project) did you feel frustrated at any time?
   - Could you describe the sort of situation in which this occurred?
   - Did you feel those about you were similarly frustrated?
   - did you feel that you were being 'thrown in at the deep end', i.e. a 'sink or swim' situation?
   - How did you feel about that.

41. If someone was contemplating taking this course what advice would you give to them?
   - What things would you want to know about them before you suggested for them to do it or not?
   - Once they decide to do it what further guidance would you give them?

42. Are there perhaps some other points that you would like to add, that haven't been brought out by the questions so far concerning the project?

*****
Appendix J. Interview Transcripts
Name of interviewee: Student 4

Group Designation: Lecture-lab

Background information:

1. Interviewer: Could you briefly tell me why you taking the CSED-420 course?
   Student 4: Why? Because I felt some of the applications could be useful in a library situation.

   Interviewer: Is it part of any particular program?
   Student 4: I'm doing a diploma in Library education.

   Interviewer: And what year are you in?
   Student 4: Fifth.

   Interviewer: And this...
   Student 4: It's a one year beyond a degree.

   Interviewer: And is this a compulsory or an elective?
   Student 4: It's an elective.

   Interviewer: So it's an elective, O.K.

2. Interviewer: So could you please describe the work you presently do?
   Student 4: I teach.

3. Interviewer: And...
   Student 4: Elementary library, grades 1 to 7, K - 7.

   Interviewer: And you have been there for?
   Student 4: Oh, God... (laughing)

   Interviewer: Yeh, O.K., that's O.K., have been there for a while.
   Student 4: 20.

   Interviewer: And you are not considering any big career change into computing?
   Student 4: No. (much laughter)

3. Interviewer: So you have done 20 years and you are in the elementary school.

4. Interviewer: Could you... so you have done 20 years and you are in the elementary school.
   Student 4: Yes, Early childhood education. Language-arts.

5. Interviewer: And um, could you describe your present involvement in computers, or have you any involvement at the moment?
   Student 4: Intro to, intro to.

   Interviewer: Intro to, O.K. And you are looking to put them into the library perhaps?
   Student 4: Yep.

   Interviewer: O.K.

   Student 4: The school has just purchased Macintoshs.

   Interviewer: Your lucky, how many?
   Student 4: I'm not sure, I think ten, and I brought one myself two weeks ago to learn.

   Interviewer: An SE?
   Student 4: No, a Macplus, the school has some SEs.

   Interviewer: You have done well then.

6. Interviewer: So you see yourself as being increasingly involved with computers?
   Student 4: I would like to.

   Interviewer: And in what other way than the library, any other way?
   Student 4: In general, in plugging the university course work, time savers.

   Interviewer: Right.

   Student 4: Household applications.

7. Interviewer: O.K. What is your present access to computers, availability?
   Student 4: I have my own.

   Interviewer: One at home. And have access to them at work now. And you actually own one now, O.K.
8. Interviewer: Could you please describe your previous programming experience?
Student 4: None.

Interviewer: Have you done summer or inter-session courses before?
Student 4: Yes.

Interviewer: How many?
Student 4: Oh, . . .

Interviewer: Been a few?
Student 4: Yeh, I suppose, oh, probably four. How many different courses?
Interviewer: That's O.K.
Student 4: Yeh, easily five.

Interviewer: And how do you find doing summer or inter session courses?
Student 4: Technically its fine, this one is not.

Interviewer: Ah ha, so why is this one not so fine?
Student 4: Highly frustrating.

Interviewer: Ah ha . . .
Student 4: Too much, too fast, no absorption time.

Interviewer: And do you find that on the other courses?
Student 4: I have never.

Interviewer: I think you have answered the next question 'How do you feel about having to do CSED-420 in three weeks'? Would you prefer to take this course over six weeks or a thirteen weeks?
Student 4: Oh, thirteen.

Approaches to Learning:

11. Interviewer: In general, how would you describe the way that you approach things in life? Are you a traditional person, conservative, radical, . . .
Student 4: Probably tending to become conservative but not . . .
Interviewer: Totally?
Student 4: Not totally.

12. Interviewer: Could you briefly describe the sort of person you think is most suited to computers?
Student 4: Somebody who thinks in mathematical . . . patterns.

Interviewer: Do you think you are a 'computer-type' person?
Student 4: I think I could be.

Interviewer: Ah ha.

Interviewer: And do you think the lecturer and the course T.A. are 'computer-type' people?
Student 4: I'd have to say yes.

13. Interviewer: Do you consider yourself a 'sciences' or 'humanities' type person?
Student 4: Humanities.

Interviewer: And if I was going to watch you at work how could I tell, if I was just observing you?
Student 4: Would there be any way that I could tell that you were a humanities type person?

Interviewer: I'm not thinking in logical fashions, I suppose.

Student 4: Is that right?

14. Interviewer: For you, what is the most efficient way to learn things?
Student 4: Probably a combination of lectures, examples that have (pause), working on examples that have some place to check for accuracy, check for more efficient methods of doing something.

Interviewer: How have you found the answers being supplied in computing?
Student 4: Zilcho.

Interviewer: Zilch, O.K. So how do you know whether things are right or otherwise?
Student 4: Certainly some of the programs will work out but that's not saying that they are the most efficient method.

Interviewer: Do you think your method, best way of learning, is reflected in your own teaching style?
Student 4: Um, no because I try to cover, I try to hit all bases.

Interviewer: What do you mean by all bases?
Student 4: For kids with other learning styles.
Interviewer: Ah ha.

15. Interviewer: If someone said you were a high achiever, how would you respond?
Student 4: I guess so, which is why I am ready to leave now. (laughing)

16. Interviewer: Are you the sort of person who does assignments ahead of time or do you tend to leave them to the last minute?
Student 4: That really depends on what the other workloads are at the time, but I try to do them ahead.

17. Interviewer: When approaching a new topic, do you prefer to be given defined goals or would you rather experiment with the topic yourself first?
Student 4: Again I think that depends on what it is. In some applications I would like a defined.

18. Interviewer: When given set assignment work, would you rather a specific assignment or to be given free choice? For the assignment?
Student 4: Right now I am appreciating the set assignments.

19. Interviewer: O.K. Do you like being given a letter grade or a numerical score on tests and assignments?
Student 4: Doesn't matter.
Interviewer: So there is no difference to you.

20. Interviewer: Do you consider yourself a factual/analytical type of person or one given to the approaching topics from a broader and more theoretical framework? That's a mouthful!
Student 4: Probably more factual.
Interviewer: More factual.
Student 4: Tending to be.
Interviewer: Would you prefer that approach in all courses you do? The factual type of approach?
Student 4: No, definitely not, there are some areas that I would strongly prefer the theoretical.
Interviewer: So it would vary?
Student 4: Mhh mm.

21. Interviewer: During the class, would you prefer the teacher to provide the organization for you or would you rather have freedom to structure your own learning?
Student 4: A combination.
Interviewer: And in CSED, are you happy with the combination?
Student 4: I beg you pardon?
Interviewer: In this course, are you happy with the combination of teacher organization and structure your own?
Student 4: Technically yes, in effect on this course, no.
Interviewer: Is that because of the three week time limit, do you think?
Student 4: Partially because of that, um, I think that its in that part that my frustration level builds, I am looking probably for more set out, but going into the lab session I would like to be working on things that I can again check out for accuracy and more efficient ways of working.

22. Interviewer: Ah ha. Are you the sort of person who seeks help or did you tend to 'nut' things out for yourself?
Student 4: I seek help but I also like to work it out for myself.
Interviewer: O.K. Where would you seek help then?
Student 4: Probably from the instructor, the T.A.
Interviewer: Anywhere else?
Student 4: I think there is a lot of value gained in student interaction.

23. Interviewer: What environment best suits your learning then: lecture, lab, tutorials, seminars, how do you feel about any of those? A preference either way?
Student 4: I don't think I do, somewhere in there, there has to be a combination of the lab and I think a lot of it depends on the instructor. The instruction that's involved.
24. Interviewer: Do you prefer quite environments for learning or do you, for instance, do you like listen to music while you study?
Student 4: Quiet.
Interviewer: Do you find yourself easily distracted or can you cut yourself off completely from what is going on around you?
Student 4: I can totally cut myself off.
Interviewer: You can?
Student 4: I almost missed the exam today.
Interviewer: Really, where were you?
Student 4: I was at home at a quarter after one.
Interviewer: Holy mackerel.
Student 4: There were no police on route.
Interviewer: Yeh, you did well.
25. Interviewer: In this course, would you prefer to work on your own, 1 to 1 to a computer or would you rather be in, with a partner in a group of say, 2 to 1 on a computer?
Student 4: 1-1
Interviewer: And any particular reasons why?
Student 4: Again I wanting to work things out on my own.
Interviewer: O.K. Did you seek much help from other class members?
Student 4: Yes, I would say so, initially until I felt that the gap between what they knew and what I knew was so much greater ...
Interviewer: Is that right.
Student 4: I bowed out.
Interviewer: Did you receive that help from class members? How did you find they responded?
Student 4: Oh, very willingly.
Interviewer: Yeh, there a pretty good bunch actually. You are lucky you have got such a good group.
26. Interviewer: In general, did you find other people around you a help, neutral or a hindrance to your learning?
Student 4: A help.
27. Interviewer: Did you think there was much interaction between the members of the CSED-420 group?
Student 4: Yes.
Interviewer: And was the interaction helpful to you?
Student 4: Mhh hmm.

The project:

28. Interviewer: How did you feel about being asked to participate in the project?
Student 4: Fine, I think it's worthwhile.
29. Interviewer: Do you recall the Group Embedded Figures Test? Out of a possible score of eighteen, how many do you think you drew correctly?
Student 4: Oh, gosh, go for 15.
30. Interviewer: You were put in the Lecture Style group. Could you describe to me your feelings about being put in that particular group?
Student 4: I had no strong feelings one way or the other.
Interviewer: Right. And if you had free choice which would you opt for lecture or lab style?
Student 4: Lab.
Interviewer: In the lecture group was it what you expected it to be?
Student 4: ... (uncertain pause) ... Yes.
Interviewer: Do you want to add anything on to that?
Student 4: Yeh, there was more interaction in your lecture sessions, than there have been with our normal instructor.
Interviewer: Interaction with the instructor?
Student 4: Student interaction.
Interviewer: Ah, right ...
Student 4: But that's a rapport, that's a part of the rapport that you have built with the group.
31. Interviewer: At any stage did you feel like opting out of the project?
Student 4:  Yes. (laughing)
Interviewer:  And why or when did that happen?
Student 4:  Oh about day three and again on day . . .
Interviewer:  And the reasons for that, was it the . . .
Student 4:  Frustrated with my own success, I guess.
Interviewer:  But you stuck to it, eh.

32. Interviewer:  Could you describe how you felt about the work load expected of you in this course?
Student 4:  I think if I understood it better I don't think the workload would be too bad.
Interviewer:  And how do you feel about it if you don't understand it . . ., its too much?
Student 4:  Its heavy!
Interviewer:  O.K.
Student 4:  I actually don't think its heavy but I don't have any other courses, for anybody else I
think it could be heavy.
Interviewer:  Has the course been a lot of work, the same, or less than other three week courses you
have taken?)
Student 4:  Fairly heavy.
Interviewer:  What about, did you find time 'dragging' or is it flying by'?
Student 4:  Flew by.

33. Interviewer:  What did you think about the course materials, handouts and references, etc.?
Student 4:  The handouts were helpful as far as they went . . .
Interviewer:  What about the textbook?
Student 4:  It was fine, but again there were no . . ., there were exercises for you to work on, which was great, but there was no solution set.
Interviewer:  Solution sets?
Student 4:  There were but you found them accidently in the chapters.
Interviewer:  The work sheets, how did you find them? The work sheets themselves?
Student 4:  I've enjoyed them.
Interviewer:  More or . . ., would you like more or less?
Student 4:  Um, probably more given a longer period of time, more work sheets but also going through each problem piece by piece regardless of whether everybody worked through it or not.
Interviewer:  How did you find the notes given on the board during the project period? I don't mind how you answer this!
Student 4:  They were fine, they were O.K.

34. Interviewer:  So, out of all the materials which did you find the most useful?
Student 4:  The text.
Interviewer:  The text.

35. Interviewer:  Did you use other resources beyond these, the ones that we have listed?
Student 4:  I sought out a friend as far as . . . picking up what I was doing.
Interviewer:  How did you find the programming quizzes? Not a good time to ask!
Student 4:  (Laughing) Frustrating. First of all the time constraints, being unsure of material, knowing that material wasn't being absorbed.

36. Interviewer:  Could you please comment on whether or not you felt the instructional component of the lessons as presented made things unnecessarily complicated or perhaps overly simple?
Student 4:  Not unnecessarily complicated and I would say they were overly simplified, no at times they have over simplified.
Interviewer:  Right, and when did you find them complicated?.
Student 4:  Um, when you are jumping into sections of a routine and assuming prior knowledge.
Interviewer:  Right.

37. Interviewer:  Do you feel you have been successful in this course so far?
Student 4:  You are asking me? Some success.
Interviewer:  So, its not been a waste of time.

38. Interviewer:  During the lessons you have expressed that you have felt frustrated so I think I can pick the rest of this up from the notes.
Student 4:  Make sure it will have applications to what you are wanting.
Interviewer:  Do you think this course will have applications . . . ?
Student 4: I think it will for me.
Interviewer: Are there perhaps some other points that you would like to add, that we haven't been brought out in this brief interview, concerning the project?
Student 4: The project itself.
Interviewer: Or in general.
Student 4: I would like feedback on the figures test.
Interviewer: Can Do. Anything else?
Student 4: Some feedback on your masters.
Interviewer: No worries.

*****
Name of interviewee: Student 5

Group Designation: Guided Self-discovery.

Background information:

1. Interviewer: Could you briefly tell me why you are taking the CSED-420 course?
   Student 5: I am finishing my degree but I don't need this course.
   Interviewer: Out of interest?
   Student 5: Yes
2. Interviewer: Have you taught before?
   Student 5: Just sub'ing
   Interviewer: Right.
   Student 5: And practicums
3. Interviewer: Secondary or elementary?
   Student 5: Some of both.
   Interviewer: How much teaching have you done then in the elementary, secondary and the sub'ing, very much?
   Student 5: I did 19 days in an elementary class straight, secondary I have only sub'ed a couple of days.
   Interviewer: Are you looking at any career change beyond teaching?
   Student 5: Not yet, not at the moment. Let me get my teaching degree first.
4. Interviewer: What are you subject concentrations?
   Student 5: My concentration is in special ed.
   Interviewer: Really that's interesting.
5. Interviewer: Could you describe your present involvement in computers.
   Student 5: Just this course, that's it.
6. Interviewer: Do you see yourself as being increasingly involved with computers?
   Student 5: Yes.
   Interviewer: In what ways do you anticipate?
   Student 5: I am involved in the Quesnel Pageant society and so I would, we would use computers to help us.
   Interviewer: Uh huh...what is that?
   Student 5: A Pageant Society?
   Interviewer: Is it in Vancouver?
   Student 5: No, it's in Quesnel...Quesnel is where I come from.
   Interviewer: Oh, Quesnel is a city, ah, right.
   Student 5: Yes.
   Interviewer: That's what I was wondering. The, um, and so you use computers for organizing that and registrations or whatever?
   Student 5: No, we've used computers but we don't have one, what usually happens is someone who is going to school, who ran the year before or something, goes in and types up all our forms on the computer.
   Interviewer: Ah, yes, got you.
   Student 5: We have form letters and we have forms and we have letters.
7. Interviewer: Where do you have access to computers at the moment?
   Student 5: I don't.
   Interviewer: Do you intend to buy one in the future?
   Student 5: Well, the pageant society is looking at buying one.
   Interviewer: Ah..(response unclear).
   Student 5: Yes.
8. Interviewer: Could you please describe your previous programming experience?
Student 5: I was enrolled in computer studies Ed-217 in 1983.
9. Interviewer: Have you done summer or inter-session courses before?
Student 5: Yes.
Interviewer: Many of them?
Student 5: Just one.
10. Interviewer: How do you find doing three week courses?
Student 5: I... fast... with three...
Interviewer: Ridiculous?
Student 5: If you blink you 'll miss it!
Interviewer: Ridiculous? Yes, so how do you find doing CSED-420 in three weeks?
Student 5: A lot of work.
Interviewer: Would you rather it over a six week or a thirteen week period.
Student 5: I would rather do it over thirteen weeks.

Approaches to Learning:

11. Interviewer: How would you describe the way you approach things in life? Are you conservative, or are you experimental, or how do you deal with things?
Student 5: (laughing) What does that have to do with this?
Interviewer: That's a good question.
Student 5: Umm... Later on when we talk about whether people are computer type people or not. Or are you just a very rational person or, ...?
Interviewer: Yep.
Student 5: Or impulsive?
12. Interviewer: Could you briefly describe the sort of person you think is most suited to computers?
Student 5: The sort of person who is suited?
Interviewer: Uh hu...
Student 5: Anybody.
Interviewer: Right, so no particular characteristic would strike you...
Student 5: Persistent, ... um, well logical.
Interviewer: Uh hu, do you think that the lecturer and course T.A. are 'computer-type' people?
Student 5: Well... why else would they be doing this, I mean?
13. Interviewer: Do you consider yourself a 'sciences' type person or a 'humanities' type person?
Student 5: Both.
Interviewer: Uh hu... by watching you work, how could I tell that you are this type of person, or what are some of the characteristics I would see you when I was watching you work?
Student 5: Um, order...
Interviewer: Right...
Student 5: A fairly methodical fashion.
14. Interviewer: For you, what is the most efficient way to learn things?
Student 5: Do it as you learn it.
Interviewer: Do you think this method of learning is reflected in your own practicums and teaching experience, teaching style?
Student 5: Not enough.
15. Interviewer: If someone said you were a high achiever, how would you respond to that?
Student 5: How so?
Interviewer: High performing, work hard...
Student 5: Oh, yes.
16. Interviewer: Are you the sort of person who does assignments ahead of time or do you tend to leave them to the last minute?
Student 5: I tend to leave them until the last minute.
Interviewer: And why do you think that is?
Student 5: Because I have a hard time getting motivated, once I get going then I'm OK.
17. Interviewer: When approaching a new topic, do you prefer to be given defined goals or would you rather experiment with things first?
Student 5: Ah, defined goals.
Interviewer: What is it that helps you having defined goals?
Student 5: I then know what to aim for.

18. Interviewer: With assignment work would you rather have a specific assignment or to be given free choice?
Student 5: Well with a specific assignment then you know exactly what he wants.
Interviewer: Mm, hmm . .
Student 5: But its nice to have some choice.
Interviewer: Right. Well, with the assignments that were set for this course, how did you feel about those?
Student 5: I think a few of them could have been a little clearer.
Interviewer: Uh hu . .

19. Interviewer: Uh hu, do you like being given a letter grade or a numerical score on tests and assignments?
Student 5: A numerical score.
Interviewer: Why is that?
Student 5: You get a better idea of the range.

20. Interviewer: Do you consider yourself a factual/analytical type of person or one given to the approaching topics from a broader and more theoretical framework?
Student 5: A factual type person.
Interviewer: Would you prefer this approach in all courses?
Student 5: No, but CSED was factual.

21. Interviewer: During the class would you prefer the teacher to provide the organization for you or would you rather have freedom to structure you own learning?
Student 5: When we are working on the computers or . . . ?
Interviewer: Just the whole class . . .
Student 5: I liked the organized worksheets. I like it ordered like they were.
Interviewer: And you were happy to have the organization in place for you?
Student 5: Yes.

22. Interviewer: Are you the sort of person who seeks help or did you tend to 'nut' things out for yourself?
Student 5: Like to nut things out first, and then if I absolutely need help then I seek help.
Interviewer: Who from, or Where did you seek help?
Student 5: Who ever was available.

23. Interviewer: What environment best suits your learning? Lectures, labs, tutorials or seminars?
Student 5: I learned the most in the lab.
Interviewer: Right, which would be your preference?
Student 5: What is the difference between tutorial or seminar?
Interviewer: A seminar is when you all sit around a table and talk, a tutorial might be a small group where someone was teaching you something.
Student 5: Then labs or tutorials over seminars.

24. Interviewer: Do you prefer quite environments for learning or do you, for instance, do you like listen to music while you study?
Student 5: No, I don't mind some noise.
Interviewer: So do you find yourself easily distracted or can you cut yourself off completely from what is going on around you?
Student 5: To start off I am very easily distracted, but once I get underway I'm O.K.

25. Interviewer: Would you prefer to work 1 on 1 with a computer or be with a partner with 2 on 1 to a computer?
Student 5: The first day I worked with someone else and it was really helpful.
Interviewer: Mm mm.
Student 5: But, I think you might tend to, ah, depend on someone else, with one on one with a computer and nobody else then you sort of have to figure it out for yourself sometimes.
Interviewer: Hmm, ah, did you seek much help form other class members?
Student 5: A few, yeh, sometimes.
Interviewer: And did you receive much help?
Student 5: Yeh.
Interviewer: And what sort of help were you able to get from the other class members?
Different ways of doing things, or... (remainder of sentence unclear).

O.K., in general, did you find other people around you a help, or were they..., was it neutral or a hindrance to your learning?

Oh, never a hindrance.

In what way did you find people helpful?

If you were stuck you could seek advice... If you found something that you couldn't figure out at all, some times X (student name given) and Y (student name given) would show me what they had gotten and I'd look at that, and try and figure it out.

Did you think that there was much interaction between the members of the CSED-420 group?

Like the students?

Mh, hmm...

Some, yes...

Was it very helpful for you?

Yeh.

Do you think the class should have been organized to allow greater interaction?

Possibly.

How did you feel about being asked to participate in the project?

Didn't mind.

In the Group Embedded Figures Test, out of a possible score of eighteen, how many do you think you drew correctly?

18

O.K.

In the lecture, you were put in the lab group weren't you? You were put in the lab group. How did you feel about that?

I was glad.

If you had free choice what would you opt for?

Lab.

Why in particular?

Because lectures are very boring.

Hmm... (pause)

More useful to work on the computer, although there were times when I felt I could have had a little more instruction before going to the computer.

Were there any times in particular?

Well, with the GOSUBS, I didn't realize exactly how they worked. I was getting the answers. I was getting the right outputs, but the way that I was doing it wasn't maybe the best, most elegant way to do it. Needed more checking to see... (response unclear)... A lot of the exercises I modified on my own, like if you just did the basic exercise, you didn't learn a whole lot.

Hmm... (pause)

But if you tried to improve on that...

Yes?

I found that is when I learned the most.

Yes, oh, was the lab style what you expected it to be?

Yes.

At any stage did you feel like opting out of the project?

No, except filling out the forms. You would go to head out the door and then remember it, oh no... .

Could you describe how you felt about the work load expected of you in this course?

Oh, it was O.K.

Mhh, hmm. Do you think you spent an excessive amount of time on this subject?

On this course?

Yes.

Yes.

Has time 'dragged by' or 'fly by'?
33. Interviewer: Now, what did you think about the course materials, used for the course?
Student 5: The handout sheets were a good idea. The example disk was a great idea.
Interviewer: In what way?
Student 5: I would, um, copy a program from it, and then work on it, change things and play around with it.

34. Interviewer: What about the text book?
Student 5: I haven't really used it, but I have used it from time to time as a reference.
Interviewer: The work sheets themselves?
Student 5: The exercises?
Interviewer: Yes.
Student 5: I thought they were good but some of the exercises, weren't specific enough exercises, didn't . . . (response unclear).
Interviewer: What about the example problems sheet, the ones I gave to you in the lab, what were they like?
Student 5: Oh, yeh, well that was really good.
Interviewer: Did you learn a lot from them?
Student 5: . . . (response unclear)

35. Interviewer: Of all the things, which did you think you learnt the most from?
Student 5: I learnt the most by doing the work sheets.

36. Interviewer: Did you use other resources beyond these?
Student 5: Mainly looking at other peoples programs.

37. Interviewer: How did you find the programming quizzes?
Student 5: I found the second one a lot harder than the first one, you don't have enough time to think, if you don't get the answer right away.

38. Interviewer: Could you tell me about how you felt about the the instructional component of the lessons. Were things unnecessarily complicated or perhaps overly simple?
Student 5: Well, they weren't unnecessarily complicated.
Interviewer: Were there any times when you felt totally at sea?
Student 5: Yes.
Interviewer: When was that.
Student 5: Um, I think the second or third day.
Interviewer: What about during the project period, or was that during the project period.
Student 5: Yes, I think in that one program with all the GOSUBS, but less and less as I went along.

39. Interviewer: Do you feel you have been successful in this course so far?
Student 5: Yes.
Interviewer: And how do you judge that?
Student 5: My test scores.
Interviewer: During the lessons did you feel frustrated at any time, other than the GOSUBS?
Student 5: No, I don't think so.

41. Interviewer: If someone was contemplating taking this course what advice would you give to them?
Student 5: It takes a lot of time, its not a nine to five course, more like overtime.

42. Interviewer: Is there anything else that I haven't asked in the questions that you think is worth commenting on about the project or about the course in particular?
Student 5: The book cost too much!

*****