Distribution Agreement

In presenting this thesis as a partial fulfillment of the requirements for a degree from Emory University, I hereby grant to Emory University and its agents the non-exclusive license to archive, make accessible, and display my thesis in whole or in part in all forms of media, now or hereafter now, including display on the World Wide Web. I understand that I may select some access restrictions as part of the online submission of this thesis. I retain all ownership rights to the copyright of the thesis. I also retain the right to use in future works (such as articles or books) all or part of this thesis.

Amy Shannon February 25, 2014

Live-Coding in Introductory Computer Science Education

by

Amy Shannon

Valerie Summet
Adviser

Mathematics and Computer Science Department

Valerie Summet

Adviser

Tom Bing

Committee Member

Bree Ettinger

Committee Member

2014

Live-Coding in Introductory Computer Science Education

By

Amy Shannon

Valerie Summet

Adviser

An abstract of
a thesis submitted to the Faculty of Emory College of Arts and Sciences
of Emory University in partial fulfillment
of the requirements of the degree of
Bachelor of Sciences with Honors

Mathematics and Computer Science Department

2014

Abstract

Live-Coding in Introductory Computer Science Education

By Amy Shannon

Live-coding, an active learning technique in which students create code solutions during class through group discussion, is an under-used method in computer science education. However, this technique may produce greater learning gains than traditional lectures while requiring less time and effort from the instructor. We begin with a discussion of active learning techniques in STEM disciplines and then present a study to evaluate this instructional method in introductory Computer Science courses. While our results were inconclusive, we discuss several interesting and positive trends related to our live-coding results that deserve further investigation.

Live-Coding in Introductory Computer Science Education

By

Amy Shannon

Valerie Summet

Adviser

A thesis submitted to the Faculty of Emory College of Arts and Sciences
of Emory University in partial fulfillment
of the requirements of the degree of
Bachelor of Sciences with Honors

Mathematics and Computer Science Department

2014

Table of Contents

List of Tables

List of Figures

Chapter 1 : Introduction

The traditional introductory computer science course is typically a student's first course in programming. Students in this course learn to write code and to apply basic coding principles and concepts. Lectures for this course are generally filled with pre-coded examples. Students often spend the majority of classroom time watching the professor explain a problem then present the code that gives a perfect solution to that problem. However, there may be a better teaching method for students who are learning their first coding language.

Other disciplines which strive to develop problem solving skills, such as physics and math, have introduced active learning techniques into their classes with great success. An active learning technique that could greatly benefit computer scientists is **live-coding**. The live-coding technique of problem solving starts with a blank text file. The students generate a code solution, and iteratively test and revise that solution, through guided group discussion during class. This process allows students to practice generating code, testing code, understanding and solving errors, and finding optimal code solutions. Live-coding gives students an interactive educational experience, with little additional effort for the professor.

We hypothesize that live-coding in the classroom produces higher learning gains for computer science students than traditional pre-coded examples. In this thesis, I report on a series of experiments designed to test this hypothesis. In Chapter 2, I present a literature review of instructional techniques, active learning techniques both in general and as applied to computer science, and previously conducted research on live-coding. In Chapter 3, I present the methodology used for this series of experiments. In Chapter 4, I present the results, which are discussed in Chapter 5. Chapter 6 provides a discussion of conclusions and future work within this research topic.

Chapter 2 : Literature Review

In this chapter, I discuss work related to several areas relevant to this thesis. In particular, I examine active learning in computer science and in other disciplines (Section 2.2), live-coding (Section 2.3), and general instructional techniques relevant to this study (Section 2.1).

2.1 Instructional Techniques

2.1.1 Flipping the classroom

Traditionally, STEM professors spend class time for lecture and give students time to practice or actively learn outside of class. Flipping the classroom is a technique in which professors "flip" the traditional division of time. Students watch or read lecture content outside of class and instead spend time in class engaged with the professor in an active learning environment. While this idea has existed in the humanities for decades (e.g. analyzing books read outside of class rather than discussing plot), recently STEM faculty have received recognition for adopting the practice. The New York Times [1], The Chronicle of Higher Education [2], and Science [3] all published articles explaining and advocating flipped classrooms in the past five years.

Variations of this technique have been shown to produce dramatic learning gains and increased student engagement. In [3] , Mazur altered his introductory physics classes to focus on interactive "clicker questions", rather than his previous method of reading from prepared lecture slides. Students became more engaged and gained better mastery of material when given the time to solve and discuss concepts during class. Deslauriers, Schelew, and Wieman compare, in [4], the effects of lecture versus flipping the classroom in large-scale physics classes. They found that students learned more, attended more classes, and were more engaged in the flipped classroom,

even though the lecture was taught by an experienced instructor and the flipped class session was led by a trained but inexperienced instructor.

During this study, we utilized the "flipped classroom" model to ensure ample time during class for coding examples, active learning techniques, and quizzes. We created videos of our lecture content for students to watch prior to attending class in addition to completing assigned textbook readings.

2.1.2 Frequent Testing

Testing has been shown to aid memory more than further studying ([5], [6]). Researchers have applied this knowledge to show frequent testing can improve performance in the classroom. Researchers at the University of Texas recently found that classroom quizzes at the beginning of each class produced better performance on midterm exams and higher attendance rates [7].

We utilized this technique by giving eleven in-class quizzes throughout the semester, in addition to the two midterms and final exams always given in this course to gather data to measure learning during this study.

2.2 Active Learning

2.2.1 Active Learning Techniques in Other Disciplines

In general, active learning is "any learning activity engaged by students in a classroom other than listening passively to an instructor's lecture" [8]. In addition to this definition, Faust and Paulson also present a catalog of active learning techniques that range from individual exercises to cooperative learning strategies to general lecture activities. These techniques have been shown to increase overall retention rate, increase lab grades, and improve student attitudes

towards their subject area. Faust and Paulson point out that although these techniques are typically used in liberal arts courses, the STEM disciplines can benefit as well [8].

Many researchers in STEM disciplines other than computer science highlight the benefits of active learning environments, including increased student engagement, learning gains, and higher retention rates. In [9], Hake studied over 6000 physics students and discovered students in courses taught using interactive, flipped-classroom models had learning gains nearly two standard deviations higher than those in traditional lecture. The effects of active learning and collaborative learning in introductory chemistry courses have been studied in Wright et al. [10]. The students in the active learning section developed better reasoning and communication skills than those in traditional lecture. Springer, Stanne and Donovan analyzed STEM education research that supports the positive effects of active and collaborative learning. They found improvements in academic achievement, favorable attitudes toward learning, and continuation in STEM courses and programs [11]. These benefits are certainly desired by computer science professors. However, active learning is still a rare practice in computer science education.

2.2.2 Active Learning in Computer Science

While the general theory of active learning began gathering support in the mid 1990's, it took time before active learning was adopted by computer science professors. While professors in other disciplines were implementing active learning techniques into their courses, their endeavors required large amounts of time and organization [12]. Many computer scientists felt active learning techniques would require too much work for too little reward, as evidenced in part by the introduction of McConnell's research, which discusses the many fears professors have regarding active learning [13]. However, in the early 2000's, computer science researchers began to consider the possibility of active learning.

In 2000, Grissom and Gorp incorporated several active learning techniques into an introductory computer science course. While their techniques included using a debugging visualization tool during class coding examples, most active learning was derived from collaborative learning exercises. In the second year of the active learning redesign, they received positive student feedback from surveys asking about the effectiveness of their techniques [14]. They did not collect data to measure student learning gains.

In 2005, Briggs introduced an active learning environment for an introductory computer science lab. He redesigned the lab projects to work for groups and wrote a lab manual for the course. The collaborative techniques led to an improvement in student grades, comprehension, and satisfaction [15].

Most active learning experiments in computer science thus far have been focused on adding collaboration through group work as the active element of the course. This approach requires significantly more work for the professor, as indicated by the multiple iterations of course design by Grissom and Gorp and the lab redesign by Briggs. The amount of effort required to achieve success with active learning continues to deter computer science professors, particularly those with large classes, from attempting these changes.

In their 2005 work, Barker and Roberts highlight three specific “fine arts” practices that could be useful to computer scientists: allowing students to hear each other articulate what they learn, mentioning practical applications of theoretical principles, and requiring students to display their knowledge and solutions to peers. These three practices are addressed by live-coding. Students must articulate to their peers during class discussion to turn theoretical concepts into practical code. Barker and Roberts note that on the rare occasions the traditional CS

professors used trial and error in lecture as opposed to pre-coded examples, students were significantly more engaged [16].

This research gives us an inkling of how live-coding could have a positive effect for computer science students – it is an active learning environment that is considerably easier for the professor to accomplish than collaborative learning through group work.

2.3 Live-Coding

While much research has been done related to computer science pedagogy, very little relates to live-coding. Of the few who have explored live-coding as a teaching practice, most report positive results based on excellent qualitative data, but collect little or no quantitative data.

In 2002, Paxton published a definition of "live programming" as "the process of designing and implementing [code] in front of the class during the lecture period". Paxton used this technique in a non-introductory class to provide an active element to lecture that "captures the programming process much more effectively than canned examples ever could"[17]. Paxton's conclusions were based on student responses to a preference survey. However, his conclusions are unclear in the context of introductory courses, as novices learn differently that those with greater mastery of a topic [18].

In 2007, the research of Gaspar and Langevin supported "live-coding" in various forms as an effective teaching strategy that helps address two trends in student learning: memorization of code and code tracing for debugging purposes[19]. Live-coding is presented as a way to discourage students from memorizing code rather than understanding code, and in turn encourage students to examine code more rigorously, particularly when catching bugs. Similar to Paxton's work, the primary source of reported data was via a student survey, and no data was provided regarding student performance.

In 2012, Nasser Giacaman used extensive live-coding demonstrations to teach parallel computing, an advanced topic for undergraduate students. Impressively, students in this course are able to "correctly and efficiently multi-thread a sequential desktop application" after only four weeks. Student survey responses indicated live-coding demonstrations and analogies used in class were the most helpful in learning these difficult concepts [20].

In 2013, Rubin began to fill the need for quantitative data regarding the use of live-coding. Rubin studied four sections of an introductory computer science course taught by two professors, where each professor had a control and an experimental section. He compared the overall scores for assignments, exams, and projects and found that the two sections with live-coding examples had significantly higher scores for the coding project and no statistically significant difference for assignments or exams [21]. While Rubin makes an excellent start at filling the gap in quantitative data about live-coding, his research raises a few questions which our study strives to investigate. For example, live-coding could be more helpful for some introductory topics than others, and the short-term benefits of live-coding (i.e. for in-class quizzes) have not been studied.

2.4 Summary

Based on the research supporting active learning as an effective technique, not just in general but also in computer science, live-coding during class should yield learning gains, increased student involvement, and increased retention for students, at little inconvenience to the professor. However, live-coding research conducted thus far presents little quantitative results based on introductory computer science classes, and there are more questions to be answered about this topic.

Chapter 3 : Methodology

In this chapter, we present a series of experiments that explores the effects of live-coding for introductory computer science students on a more detailed level while controlling for previous coding knowledge. We seek to understand if live-coding has a greater impact with certain topics more than others, and if live-coding has short-term benefits (i.e. for in-class quizzes) as well as long-term advantages for students' overall learning gains by gathering quantitative data.

3.1 Procedure

3.1.1 Design

We had hoped to study two sections of the course, using one as a control group and one as an experimental group. Due to the limited number of faculty in the department, we had access to only one section of the introductory computer science course. This dilemma presented us with the problem of how to divide students sitting in the same classroom into a control group and an experimental group. We split each of three major course topics into a control subtopic and an experimental subtopic. The concepts of the control subtopic were explained using only traditional pre-coded examples, and the concepts of the experimental subtopic were explained using only live-coding examples. We gave one quiz on each subtopic, so each major topic had two quizzes, one control and one experimental, from which to compare data.

The three major topics covered were conditional execution (i.e. if statements), iteration (i.e. loops), and encapsulation (i.e. methods). As much as possible, we chose subtopics to be equivalent in conceptual difficulty and amount of material. For example, else statements account for the "other half" of the condition introduced by if statements. For loops and while loops perform the same function with different syntax. Parameters and return values are the inputs and

outputs of methods and are very similar conceptually. We alternated having the control subtopic first since the first subtopic is generally easier than the second subtopic, as they build off of each other. We recognize that there is the possibility of cross-contamination in that instruction on one subtopic could lead to improved learning on the other subtopic. However, this risk has been minimized as much as possible through the specific division of subtopics and timing of the quizzes. The only difference between the control subtopics and the experimental subtopics is that the experimental subtopics are explained using live-coding examples, while the control subtopics are explained using traditional pre-coded examples. Live-coding is the only active learning technique used in this course. The subtopics and distribution between control and experimental subtopics are as follows:

CONDITIONALS

Boolean variables; If statements	Experimental
Else statements; If else statements	Control

LOOPS

While loops	Control
For loops	Experimental

METHODS

Parameters	Control
Return Values	Experimental

In order to minimize the possibility of designing our quizzes too closely to the examples given in class, we divided the duties between the two researchers. Researcher A, who was also the professor of the course, designed the examples to be given in class. Researcher A also determined which topics would have live-coding examples and which would be traditional examples. The examples were established several weeks before each class session. Researcher B

designed the quizzes, without any knowledge of the type of example being used for each topic or the content of that example. Researcher A reviewed the quizzes only to ensure they were appropriate in content and difficulty. The quizzes were usually finalized the day the quiz was given in class, well after the examples had been determined, making it impossible for the examples to "teach to the test".

3.1.2 What Happened in Class

Lecture videos were posted online for students to watch prior to attending class on each topic. At the beginning of each 75 minute class, the instructor led a 5-10 minute question and answer period.

Students were then shown an example, beginning with a problem statement in English (no code involved). After discussing a problem solving strategy, a basic solution (in Java code) was discussed. The basic solution was then iteratively refined to a final solution, and possible errors were discussed. The example would either be a traditional pre-coded example or a live-coding example, depending on the topic of that particular class session. For a traditional example, the basic code solution and each refinement was given to the class (pre-coded) and discussed. For a live-coding example, the class would generate through discussion the basic solution and each refinement leading to the final solution.

The example typically took between 20 and 25 minutes to demonstrate. Students would then immediately be given an in-class quiz lasting no more than 10 minutes. After the quiz was collected, the class would continue as normal, often into a discussion of the quiz questions or further examples.

Students were not told the subject of the research; the difference between pre-coded and live-coding examples was never explicitly stated, nor did we ever point out that some course

topics were being explained with different types of examples. Students had after-class access to all materials written or presented in class, including code solutions and slide-show presentations.

3.2 Materials

3.2.1 Assessments

Students were given two assessments during the second week of the course: a Learning Styles Assessment and a Prior Coding Knowledge Assessment. Students were given full credit for completing these assessments, regardless of their answers. We used the Prior Coding Knowledge Assessment to determine if any students had sufficient coding knowledge before this course to be excluded from our data. We used the Learning Styles Assessment to investigate whether live-coding had a greater affect for students with particular learning preferences.

The Learning Styles Assessment was the VARK Questionnaire version 7.1, a series of sixteen multiple-answer questions designed to provide a profile of learning preferences. Learning preferences identified with the VARK Questionnaire are Visual, Aural, Read/Write, and Kinesthetic. Recently in [22], Leite conducted research confirming the validity and reliability of the VARK Questionnaire. The VARK Questionnaire is provided in Appendix B.

The Prior Coding Knowledge Assessment consisted of five multiple-choice or multiple-answer questions testing knowledge of the Java coding language and basic computer science concepts. Each question had “I don’t know” as an answer option. This assessment was designed to determine if any students had advanced coding knowledge prior to taking the course, so we could eliminate those students from our data set. We also gave the same questions as an in-class quiz at the end of the course, without “I don’t know” as an answer option. This assessment can be found in Appendix A.

3.2.2 Lecture Videos

Lecture videos were created by the course professor. These videos captured a whiteboard and the professor's voice only. The content included a basic introduction of each topic covered in the course. Videos did not contain detailed coding examples in which code was written or used to solve problems. However, some small code segments were introduced and explained, and there were limited instances of code tracing, during which the professor would outline what happened to a particular variable or value when a few lines of code were executed. Thirteen videos were made, six associated with our study topics. The six videos related to our research ranged from 12 minutes 25 seconds to 26 minutes 38 seconds in length, with an average length of 19 minutes 57 seconds. Students could access all videos online, and were expected to watch one video before each class session so the majority of class time could be spent on code examples and active learning exercises. Students watched a video for every topic in our study, regardless of the type of examples used for that topic during class. While these videos did change the amount of work students were expected to do outside of class, the addition of approximately 40 minutes per week was not substantial enough to alter their workload to be greater than the expectation for a 4 credit hour course.

3.2.3 Quizzes

Students were given 11 in-class quizzes throughout the semester. The purpose of the quizzes was to test students' knowledge on subtopics for which they had seen either traditional coding examples or live-coding examples. The quizzes were intended to be the main criteria used to determine the effectiveness of live-coding examples. Quizzes generally consisted of at least two questions, where the first question asked for basic understanding of a concept and the second asked student to demonstrate the ability to apply that concept. For example, a student might be

asked to evaluate Boolean expressions like those in Figure 3.1, then asked to determine the output and function of a code segment like the one in Figure 3.2. Many of our questions had multiple parts.

Figure 3.1 Boolean Expressions

Evaluate each expression to be TRUE, FALSE, or containing an ERROR.
Assume the program has executed the statements: x=2; y=9; z=6;

A. (x>y && y>z)
B. (x>y || y>z)
C. !(x>y)
D. (x + y > z)

Figure 3.2 Sample Code Segment

```
String s = "mystery";

for( int i = s.length()-1; i>=0; i--){
     S.O.P (s.charAt(i));
}
S.O.P("----");
for( i=0; i< s.length(); i+=2){
     S.O.P (s.charAt(i));
}
```

As much as possible, we made the two quizzes given for a particular topic match in question type and question difficulty. We used the principles of Bloom's Taxonomy to assess the comparability of questions [23]. The following paragraph, along with the outline below, provides an example of how we used Bloom's Taxonomy in creating comparable quizzes for Loops. The quizzes are provided in Appendix C.

The questions on paired quizzes have similar structure. The code a student must interpret is comparable in both length and density. Even the phrasing of particular questions is nearly identical when asking for similar tasks. When we apply Bloom's Taxonomy to these questions, we find further similarity. Consider the loops quizzes outlined below. The first question on each quiz involves understanding basic loop structure and applying that understanding to determine a

basic function of a loop. These skills fall in to the Comprehension and Application areas for Bloom. The second question on each quiz asks students to interpret a larger code segment. Students again use Comprehension and Application to determine the code's output, but a greater mastery of these skills is required here than in Question 1. Then, students are asked to write a sentence explaining the code's function – Analysis. These quizzes cover three levels of Bloom's Taxonomy in similar proportions.

While Loops Question 1

- 5 option Multiple Answer
- short code segments (1 loop; 1 variable; 4 lines long)
- determining basic code function → how many iterations of loop

While Loops Question 2

- two part short answer
- long code segment (1 loop; 2 variables; 15-20 lines long)
- determine output of entire code segment
- write a sentence to describe the code's function

For Loops Question 1

- 5 option Multiple Answer
- short code segments (1 loop; 1 variable; 3 lines long)
- determining basic code function → output of each loop iteration

For Loops Question 2

- two part short answer
- long code segment (2 loops; 2 variables; 10 lines long)
- determine output of entire code segment
- write a sentence to describe the code's function

We were primarily interested in three major topics covered in this course – conditionals, loops, and methods. These three topics were covered in two quizzes each (one for live-coding examples and one for traditional examples). We collected data primarily from these six quizzes. We also gave 2 initial quizzes to get students in the habit of watching the videos before class and

3 additional quizzes on material covered after the three major topics, for a total of 11 in-class quizzes. All quizzes are provided in Appendix C.

3.2.4 Final Exam

Eleven multiple choice questions were added to the final exam for this course. Most final exam questions test mastery of several topics within a single question. We added questions designed to test only one major topic at a time. These questions were provided in Appendix D for the thesis defense, but were excluded from the online repository as the questions will be used for future final exams.

3.2.5 Departmental Surveys

Each semester, at the end of each course, the Math and Computer Science department provides anonymous surveys for students to comment on the strengths and weaknesses of the course and instructor. These surveys are not available to the instructor until after the final course grades have been submitted. We examined the responses to these surveys for our research. The survey is available in Appendix E.

3.3 Participants

Thirty-one students were recruited based on their enrollment in a CS 1 course. Every student was given the option to participate in the study. Participation would have no effect on their course grade, nor would it involve any additional effort or time for students. Informed consent was obtained from all participants, and consent forms were not seen by the course professor until after final grades had been submitted. All students received the same instruction throughout the course.

Two students withdrew from the course mid-semester; their data is not included in our results. One student had significant prior coding knowledge and was excluded from the results. Of the remaining twenty-eight students, 12 were female and 16 male. Table 3.1 displays the distribution of students by major, and Table 3.2 displays the distribution of students by year.

Table 3.1 Student Distribution by Graduation Year

Freshmen	Sophomore	Junior	Senior	Non-Degree Seeking
4	6	7	10	1

Table 3.2 Student Distribution by Major
Students with double majors are listed in multiple categories.

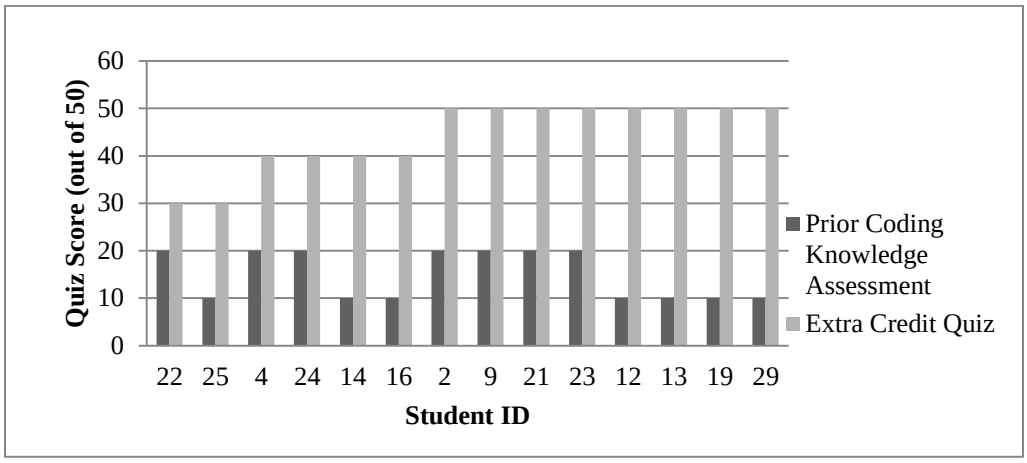
Mathematics	16
Business or Economics	13
Science (includes Psychology)	8
Computer Science	4
Humanities (includes Poly Sci)	4
Undeclared	4
Non-Degree Seeking	1

Chapter 4 : Results

4.1 Pre and Post Test

The Prior Coding Knowledge Assessment given at the beginning of the semester consisted of the same multiple choice questions as the Extra Credit Quiz given at the end of the semester. The only difference between these tests was the additional option to select "I don't know" in the Prior Coding Knowledge Assessment only. These assessments served as a pre and post test for our students. A visual representation of student scores for the Pre Test (M = 14, SD = 5, n=16) and the Post Test (M = 44, SD = 7, n=22) are shown in Figure 4.1. Students are listed in order of increasing improvement.

Figure 4.1 Pre and Post Test Comparison



4.2 Quiz Scores

We collected data for six quizzes covering three major topics of this course. Table 4.1 lists the mean quiz score, standard deviation, and number of students for each of these quizzes. This information is displayed graphically in Figure 4.2. In this figure, bars represent the mean +/- one standard deviation.

Using R, we ran the default t-test on our data, comparing the quizzes in each major topic. This test assumes unequal variance and unpaired data. We operated under a null hypothesis that the two quizzes would have no difference in mean score, with an alternative hypothesis that the means were not equal. Live coding did not have a significant effect for conditional topics ($t(42) = 2.3$, $p = .03$), for iteration topics ($t(46) = 2.0$, $p = .06$), or for encapsulation topics ($t(40) = 1.1$, $p = .3$).

Table 4.1 Quiz Scores by Topic

Quiz Topic	Mean	SD	# students
If Statements	36	6	28
Else Statements	42	9	25
While Loops	35	12	25
For Loops	41	9	26
Parameters	31	13	26
Return Values	34	8	24

Figure 4.2 Mean Scores by Quiz Topic

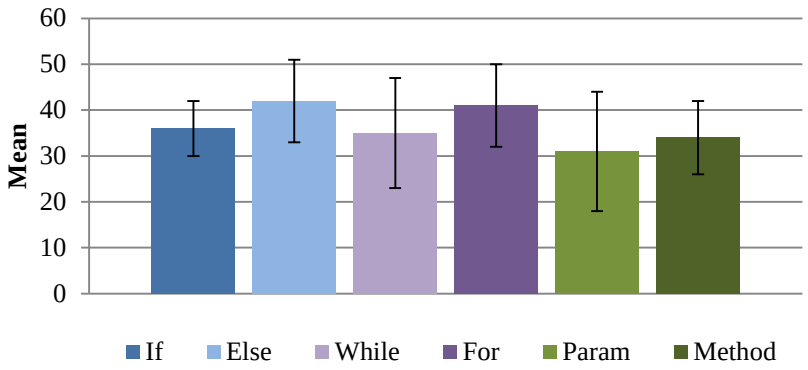
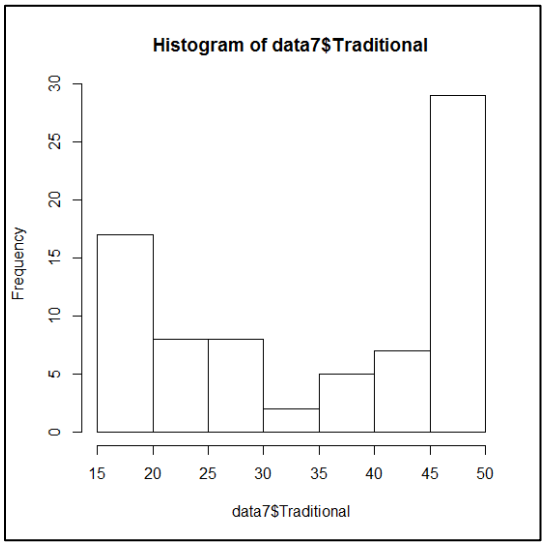
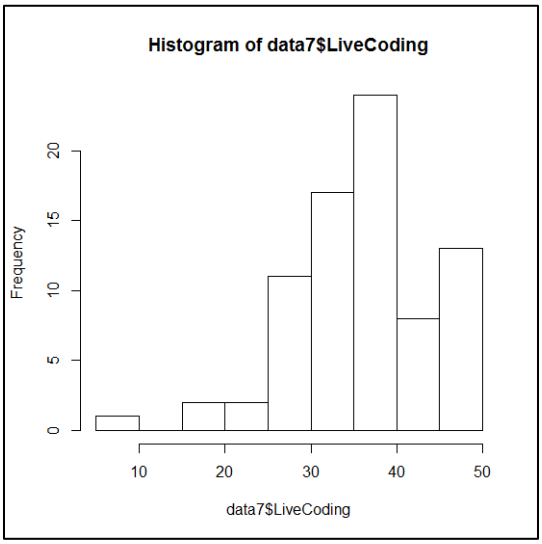


Figure 4.3 shows the differences between histograms of the scores on live-coding quizzes and traditional quizzes. These graphs represent data from every quiz throughout the course.

Figure 4.3 Histograms of Quiz Scores





Finally, we compared the effect live-coding had on the overall quiz average when gender was considered. Figure 4.4 shows this division, determined by averaging all quiz scores for a particular condition and for a particular gender to create each column or box. Note that for this and all following boxplots, the dashed lines indicate range, the circle indicates outliers, the bold line indicates median, and the box indicates quartiles. Figure 4.5 displays quiz means +/- one standard deviation. We again used the standard R unpaired t-test, with the alternative hypothesis that the mean quiz scores were not equal. Live coding did not have a significant effect for women ($t(18) = .64$, $p = .53$) or for men ($t(26) = .74$, $p = .46$).

Figure 4.4 Quiz Averages by Gender and Condition

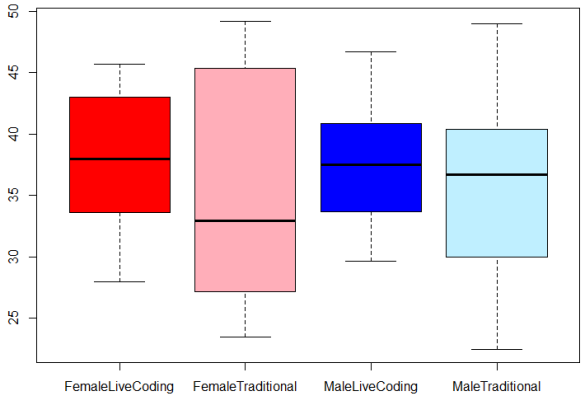
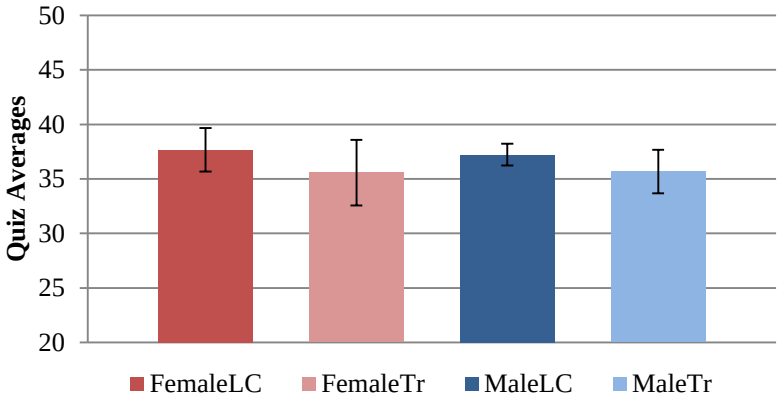


Figure 4.5 Mean Quiz Scores by Gender



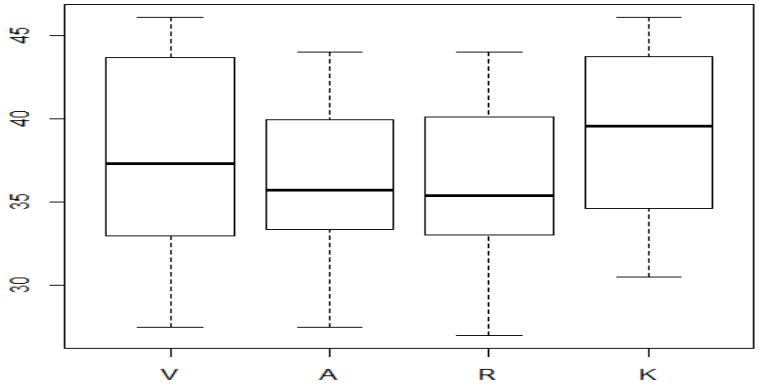
4.3 Learning Style

At the beginning of the course, we gave each student a Learning Styles Assessment to determine their preference for visual, aural, read/write, and kinesthetic learning. Table 4.2 shows the distribution of students between the learning styles, and the overall quiz average for each group. If a student were to answer this test by selecting every answer for every question, they would have a 25% preference for each learning style. Therefore, we considered a student to prefer a particular style if they chose more than 25% of their answers to be within a particular style.

Table 4.2 Overall Quiz Average by Learning Style
students with multiple preferences listed in multiple categories

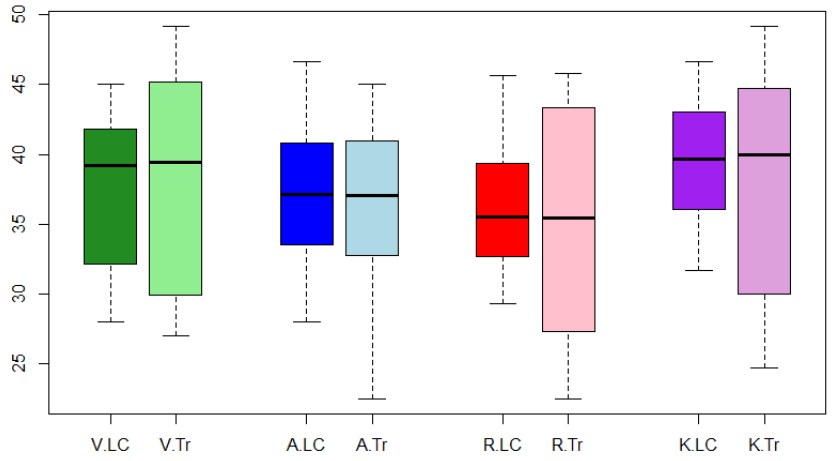
Learning Style	# Students	Overall Quiz Average
V (Visual)	8	38
A (Aural)	12	37
R (Read/Write)	14	36
K (Kinesthetic)	15	38

Figure 4.6 Overall Quiz Average by Learning Style



A visual representation of the quiz averages, shown in Figure 4.6, shows a slightly higher Quiz average for Kinesthetic learners. However, when each learning styles group is divided into the quiz average for live-coding topics and traditional example topics, there is no difference in achievement within each group (Figure 4.7).

Figure 4.7 Live-Coding and Traditional Quiz Averages by learning Style



4.4 Department Surveys

The departmental survey asks three very general questions about the strengths and weaknesses of the course and instructor (see Appendix E). The survey did not ask any specific questions related to our changes to the course.

Of the 16 surveys we received, one was illegible. Three expressed negative opinions of the course of instructor, perhaps due to other interactions between the faculty and students during this course. Of the remaining 12 surveys, 5 expressed positive opinions about the lecture videos. These students found the videos to be “helpful for studying” and even complained about the lack of videos for the last topics in the course. Two students mentioned the additional quizzes, saying the constant testing forced them to stay on top of the material. However, they would have preferred advance notification of quizzes. One student made an ambiguous comment that could be directed toward live-coding: “I found the days we actually worked with code on the computer to be particularly helpful”.

4.5 Final Exam Questions

Students answered twelve additional multiple choice questions on the final exam that asked about individual topic areas (rather than the typical final exam question that asks students to demonstrate knowledge of several topics per question). Figure 4.8 shows how many questions students answered correctly (M = 8, SD = 2), and Figure 4.9 what percentage of students correctly answered each question (M = 71%, SD = 22%).

Figure 4.8 Final Exam Scores

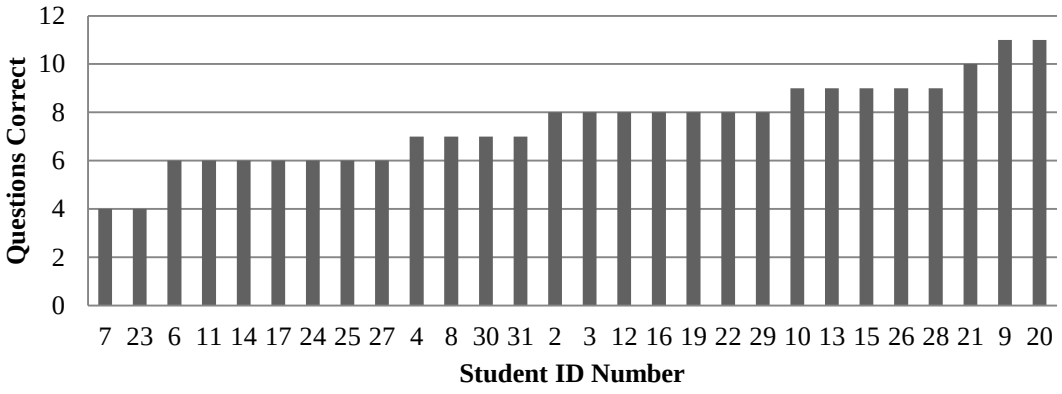
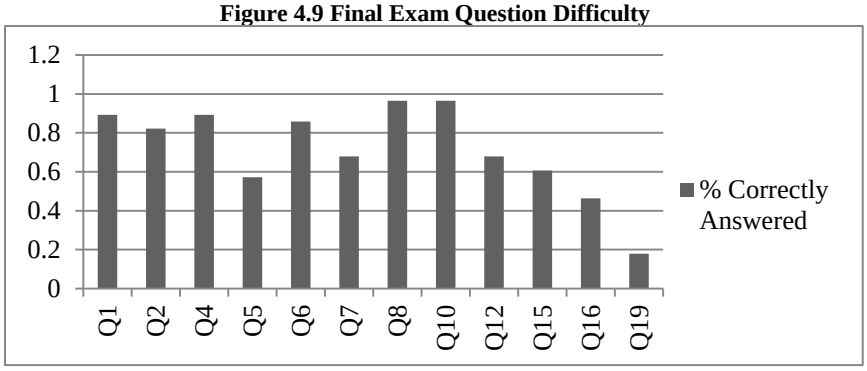


Figure 4.9 Final Exam Question Difficulty



4.6 Attendance

We kept attendance records for 16 class sessions throughout the semester. On average, students attended 89% of those classes (SD=13). Six students had attendance more than one standard deviation below the mean for these sessions. Students were not graded on course

attendance, but since quizzes were unannounced, missed classes often translated into missed quizzes.

4.7 Lecture Video Data

The lecture video host provided limited data regarding student usage of the online videos for our class. We do not have individualized data to examine by-student viewership. Therefore, these numbers may include data from our one student outlier. Table 4.3 shows how many students viewed each video, how many of those views represented repeat viewers, and what percentage of the video was completed. The mean and standard deviation for these values are also listed.

Table 4.3 Lecture Videos

Video Topic	Unique Views	Cumulative Views	Repeat Viewers	Completion Percentage
Booleans/If	25	39	14	93
If/Else	23	29	6	95
While	22	29	7	90
For	17	24	7	94
Parameters	22	29	7	85
Return Values	23	35	12	88
Mean	22	31	9	91
Standard Dev.	2	5	3	4

Chapter 5 : Discussion

Our results were largely inconclusive in determining the positive effects of live-coding. However, it is important to note that our changes to the course did no harm. In fact, some students reported benefits from the lecture videos and frequent quizzing. There was no statistical difference in quiz scores between subtopics taught with live-coding examples and those taught with traditional examples. Despite our lack of statistical significance, several of our results raise interesting questions and observations.

5.1 Analysis

5.1.1 Gender

When examining Figure 4.4 we noticed a difference in the boxplots of live-coding and traditional quizzes. The scores of traditional quizzes covered a much wider range than for live-coding quizzes, and female students had a lower median score on traditional quizzes than any other category. We then graphed the mean +/- one standard deviation to further investigate (Figure 4.5). While we did not show significant difference in mean quiz scores for live-coding, we believe the differences shown in Figure 4.4 deserve further investigation. Literature indicates that the live-coding technique may benefit students, particularly women, by creating an active learning environment that encourages collaboration and community among students. While we believe that could play a role, the specific source of this phenomenon requires further investigation. A study with access to a larger group of students, more extensive surveys, and recorded observation of student behavior in and out of class could perhaps determine more about the relationship between gender and live-coding.

5.1.2 Learning Style

Initially we were curious whether live-coding had a different effect for students with different learning style preferences. We hypothesized that this method may be more helpful to students who prefer kinesthetic learning, or even aural learning, as it increases the amount of engaging discussion time during class. However, as shown in Figure 4.7, within each group of learning style preference, there is no difference in average quiz score between the two conditions.

5.1.3 Student Preferences

Some of the changes to this course caused additional work for students. Primarily, we expected everyone to watch a lecture video before attending each class. While the amount of time was minimal (an average of just under 20 minutes per video), we expected our students to complain. We also assumed they would complain about frequent, unannounced quizzes during class that served as an encouragement to attend more classes.

Contrary to our assumptions, students found the lecture videos useful and helpful for studying. Some even complained about the lack of videos for topics towards the end of the course. In addition, students acknowledged that the frequent quizzing benefited them by encouraging students to stay on top of new information. More specific analysis of student survey responses is found in Section 4.4. We found these responses particularly encouraging given that the only prompt provided was "Comment on the strengths and weaknesses of the course and instructor".

5.1.4 Topic Distribution

Although we did not have enough data to determine statistical significance, students consistently performed slightly better on the second quiz within each topic (see Table 4.1 and Figure 4.2). One possible explanation could be that because the second subtopic builds on the first, instruction on the first subtopic improves learning for the second subtopic. Also, the first quiz always tests a completely new topic and concept, while the second quiz tests a new topic, but not a new concept. For example, while loops are the first loops these students learn. For loops are different, but the basic repetition concept still applies and has already been taught. In a study where every topic was taught with traditional examples for one group and live-coding examples for a different group, this phenomenon may not occur, or may occur to a lesser degree.

When examining the histograms in Figure 4.3, notice that live-coding quiz scores are closer to normal distribution and the traditional example quizzes exhibit a bimodal distribution.

We recognize that the limited number of students in our study severely limits the amount of certainty available from these graphs. However, we believe further investigation into this area could reveal interesting findings regarding the effectiveness of live-coding.

5.2 Limitations

Our study was limited in several key ways. We had only one section of students. This limitation caused us to split instruction between traditional examples and live-coding examples. This limitation created several issues. First, the same students were exposed to both conditions throughout the experiment. This is a threat to internal validity, as the effects of live-coding on one subtopic may have caused benefits in another subtopic. Second, only some of the subtopics were taught using live-coding examples. It could be that some subtopics are more suited to live-coding than others, but we were unable to fully investigate this hypothesis in this study. Third, we had to design instruments to measure not only across conditions, but also across topics. The choice of instrument is a threat to internal validity, as some quizzes may have been harder than others. Finally, we had a small number of students from which to collect data. The small n limited our statistical power significantly. Our ideas to improve these limitations are explored in Chapter 6.

5.3 Benefits of this Study

While the results were statistically inconclusive, this research serves as an excellent pilot study in preparation for further investigation of live-coding. We created lecture videos that can be used in future courses. Our quizzes serve as a baseline for future iterations of measuring student learning gains. Many of the limitations in our study could be removed in the future. We have discovered several interesting areas to explore in relation to live-coding.

Chapter 6 : Conclusions and Future Work

Based on active learning literature and prior live-coding research, we believe the practice of live-coding deserves further investigation. We would like to conduct a similar study with multiple sections of the introductory computer science course taught by the same professor. This alteration would allow us to have separate classrooms for our control group and experimental group. In addition to increasing the number of students involved in the study, it would also ensure every topic could be taught with live-coding and with traditional examples. Finally, this change would eliminate the challenge of trying to create comparable quizzes on different subtopics. We could also use multiple methods to evaluate student learning gains. In addition to in-class quizzes, the grades from homework, projects, and labs could also be considered. Overall, using multiple sections of the same course, or one section taught across multiple semesters, would greatly improve the investigative power of this research. Given the positive trends our study found regarding student preferences for live-coding and the possible relationship between gender and live-coding, we believe further investigation based on our pilot study as described above is a valuable research endeavor.

References

1. Fitspatrick, M., *Classroom Lectures Go Digital*, in *New York Times*. 2012.
2. Berret, D., *How 'flipping' the classroom can improve the traditional lecture*, in *The Chronicle of Higher Education*. 2012.
3. Mazur, E., *Farewell, Lecture?*, in *Science*. 2009. p. 50-51.
4. DesLauriers, L., E. Schelew, and C. Weiman, *Improved Learning in a Large-Enrollment Physics Class*, in *Science*. 2011. p. 862-864.
5. Karpicke, J. and J. Blunt, *Retrieval Practice Produces More Learning than Elaborative Studying with Concept Mapping*, in *Science*. 2011. p. 772-775.
6. Roediger, H.L. and J.D. Karpicke, *Test-enhanced learning: taking memory tests improves long-term retention.* Psychological Science, 2006. **17**(3): p. 249-55.
7. Carey, B., *Frequent Tests Can Enhance College Learning, Study Finds*, in *New York Times*. 2013.
8. Faust, J. and D. Paulson, *Active Learning in the College Classroom.* Journal on Excellence in College Teaching, 1998. **9**(2): p. 3-24.
9. Hake, R., *Interactive-engagement versus traditional methods: A six-thousand-student survey of mechanics test data for introductory physics courses.* American Journal of Physics, 1998(66): p. 64-74.
10. Wright, J.C., et al., *A novel strategy for assessing the effects of curriculum reform on student competence.* Journal of Chemical Education, 1998. **75**: p. 986-92.
11. Springer, L., M.E. Stanne, and S. Donovan, *Effects of Cooperative Learning on Undergraduates in Science, Mathematics, Engineering, and Technology: A Meta-analysis*. Research Monograph No. 11. Vol. National Institute for Science Education, Review of Educational Research. 1998, Madison: University of Wisconsin-Madison.

12. Smith, A., et al., *Introductory Biology Courses: A Framework to Support Acitve Learning in Large Enrollment Introductory Science Courses.* Cell Biology Education, 2005. **4**(2): p. 143-156.

13. McConnell, J.J., *Active learning and its use in computer science.* SIGCSE Bull., 1996. **28**(SI): p. 52-54.

14. Grissom, S. and M.J.V. Gorp, *A practical approach to integrating active and collaborative learning into the introductory computer science curriculum.* J. Comput. Sci. Coll., 2000. **16**(1): p. 95-100.

15. Briggs, T., *Techniques for Active Learning in CS Courses.* Journal of Computing Sciences in College, 2005. **21**(2): p. 156-165.

16. Barker, L., K. Garvin-Doxas, and E. Roberts, *What can computer science learn from a fine arts approach to teaching?* SIGCSE Bull., 2005. **37**(1): p. 421-425.

17. Paxton, J., *Live programming as a lecture technique.* J. Comput. Sci. Coll., 2002. **18**(2): p. 51-56.

18. Bransford, J.D., A.L. Brown, and R.R. Cocking, eds. *How People Learn: Brain, Mind, Experience and School*. 2000, National Research Council.

19. Gaspar, A. and S. Langevin, *Restoring "coding with intention" in introductory programming courses*, in *Proceedings of the 8th ACM SIGITE conference on Information technology education*. 2007, ACM: Destin, Florida, USA. p. 91-98.

20. Giacaman, N., *Teaching by Example: Using Analogies and Live-coding Demonstrations to Teach Parallel Computing Concepts to Undergraduate Students*, in *Proceedings of the 2012 IEEE 26th International Parallel and Distributed Processing Symposium Workshops & PhD Forum*. 2012, IEEE Computer Society. p. 1295-1298.

21. Rubin, M.J., *The effectiveness of live-coding to teach introductory programming*, in *Proceeding of the 44th ACM technical symposium on Computer science education*. 2013, ACM: Denver, Colorado, USA. p. 651-656.

22. Leite, W., M. Svinicky, and Y. Shi, *Attempted Validation of hte Scores of hte VARK: Learning Styles Inventory with Multitrait-Multimethod Confirmation Factor Analysis Models.* Educational Psychology and Measurement, 2012. **70**: p. 323-39.

23. Starr, C.W., B. Manaris, and R.H. Stalvey, *Bloom's taxonomy revisited: specifying assessable learning objectives in computer science.* SIGCSE Bull., 2008. **40**(1): p. 261-265.

Appendix

Appendix A: Prior Coding Knowledge Assessment / Extra Credit Quiz

1. Consider the following piece of code:

```
int x=1;
while (x!=16){
   System.out.println("The value of x is: " + x);
   //line of code here
}
```

Which of the following statements could be inserted in place of the comment to avoid an infinite loop? Choose ALL that apply.

a. x = x + 2;
b. x = x + 3;
c. x = x + 4;
d. x = x * 2;
e. x = x * 3;
f. x = x - 1;

2. The function `twist` is defined as follows:

```
public void twist(String w1, String w2){
  int L0 = w1.length();
  int L1 = w2.length();
  String tmp = w1.substring(0,1);
  w1 = w2.substring(0,1) + w1.substring(1,L0);
  w2 = tmp + w2.substring(1,L1);
  System.out.println(w1 + " " + w2);
}
```

What is the output of the function call `twist("HOW", "NEAT");`?

a. NOW NOW
b. HOW HOW
c. NOW HOW
d. HOW NEAT
e. NOW HEAT

3. Consider the following code:

```
int y = 4;
double z = 4.0;
y = y + 2;
z = y + 5.0;
```

What is the value of y after the code above executes?

a. 4
b. 6
c. 8
d. 9

e. 11

4. What is the value of the array a after the following code executes?

```
int[] a = {1,1,1,1,1,1,1};
for(int i=1; i< a.length; i++){
   if( (i%2) == 1 ){
      a[i] = a[i] + a[i-1];
   }
}
```

a. {1,1,1,1,1,1,1}
b. {1,2,2,2,2,2,2}
c. {1,2,1,2,1,2,1}
d. {1,2,3,5,8,13,21}
e. {1,2,3,4,5,6,7}

5. What does the following code print when it is executed?

```
int num = 20;
if (num <= 40) {
     System.out.println("hi");
} else if (num <= 30) {
     System.out.println("bye");
} else if (num <= 20) {
     System.out.println("sigh");
} else {
     System.out.println("cry");
}
```

a. hi
bye
sigh
cry

b. hi
bye
sigh

c. hi
bye

d. hi

e. sigh

Appendix B: Learning Styles Assessment

The VARK Questionnaire (Version 7.1)

Choose the answer which best explains your preference and circle the letter(s) next to it. **Please circle more than one** if a single answer does not match your perception.
Leave blank any question that does not apply.

1. You are helping someone who wants to go to your airport, the center of town or railway station. You would:

a. go with her.
b. tell her the directions.
c. write down the directions.
d. draw, or give her a map.

2. You are not sure whether a word should be spelled `dependent' or `dependant'. You would:

a. see the words in your mind and choose by the way they look.
b. think about how each word sounds and choose one.
c. find it online or in a dictionary.
d. write both words on paper and choose one.

3. You are planning a vacation for a group. You want some feedback from them about the plan. You would:

a. describe some of the highlights.
b. use a map or website to show them the places.
c. give them a copy of the printed itinerary.
d. phone, text or email them.

4. You are going to cook something as a special treat for your family. You would:

a. cook something you know without the need for instructions.
b. ask friends for suggestions.
c. look through the cookbook for ideas from the pictures.
d. use a cookbook where you know there is a good recipe.

5. A group of tourists want to learn about the parks or wildlife reserves in your area. You would:

a. talk about, or arrange a talk for them about parks or wildlife reserves.
b. show them internet pictures, photographs or picture books.
c. take them to a park or wildlife reserve and walk with them.
d. give them a book or pamphlets about the parks or wildlife reserves.

6. You are about to purchase a digital camera or mobile phone. Other than price, what would most influence your decision?

a. Trying or testing it.
b. Reading the details about its features.
c. It is a modern design and looks good.
d. The salesperson telling me about its features.

7. Remember a time when you learned how to do something new. Try to avoid choosing a physical skill, eg. riding a bike. You learned best by:

a. watching a demonstration.
b. listening to somebody explaining it and asking questions.
c. diagrams and charts - visual clues.
d. written instructions – e.g. a manual or textbook.

8. You have a problem with your heart. You would prefer that the doctor:

a. gave you a something to read to explain what was wrong.
b. used a plastic model to show what was wrong.
c. described what was wrong.
d. showed you a diagram of what was wrong.

9. You want to learn a new program, skill or game on a computer. You would:

a. read the written instructions that came with the program.
b. talk with people who know about the program.
c. use the controls or keyboard.
d. follow the diagrams in the book that came with it.

10. I like websites that have:

a. things I can click on, shift or try.
b. interesting design and visual features.
c. interesting written descriptions, lists and explanations.
d. audio channels where I can hear music, radio programs or interviews.

11. Other than price, what would most influence your decision to buy a new non-fiction book?

a. The way it looks is appealing.
b. Quickly reading parts of it.
c. A friend talks about it and recommends it.
d. It has real-life stories, experiences and examples.

12. You are using a book, CD or website to learn how to take photos with your new digital camera. You would like to have:

a. a chance to ask questions and talk about the camera and its features.
b. clear written instructions with lists and bullet points about what to do.
c. diagrams showing the camera and what each part does.
d. many examples of good and poor photos and how to improve them.

13. Do you prefer a teacher or a presenter who uses:

a. demonstrations, models or practical sessions.
b. question and answer, talk, group discussion, or guest speakers.
c. handouts, books, or readings.
d. diagrams, charts or graphs.

14. You have finished a competition or test and would like some feedback. You would like to have feedback:

a. using examples from what you have done.
b. using a written description of your results.
c. from somebody who talks it through with you.
d. using graphs showing what you had achieved.

15. You are going to choose food at a restaurant or cafe. You would:

a. choose something that you have had there before.
b. listen to the waiter or ask friends to recommend choices.
c. choose from the descriptions in the menu.
d. look at what others are eating or look at pictures of each dish.

16. You have to make an important speech at a conference or special occasion. You would:

a. make diagrams or get graphs to help explain things.
b. write a few key words and practice saying your speech over and over.
c. write out your speech and learn from reading it over several times.
d. gather many examples and stories to make the talk real and practical.

Appendix C: Quizzes

Quiz 1: Booleans and If Statements

1. (20 points) Evaluate each expression to be TRUE, FALSE, or containing an ERROR. Assume the program has executed the statements: x=2; y=9; z=6;

A. (x>y && y>z)
B. (x>y || y>z)
C. !(x>y)
D. (x + y > z)

2. (30 points) Prof. C.S. Iscool is assigning letter grades to her class. A student receives an "A" if they score 200 points or more on their assignments. They may ALSO receive an "A" if they score 150 points or more and did the extra credit. If they didn't do the extra credit, then they receive a "B" if they scored 100 points or more, but less than 200 or a "C" if they score between 0 and 100 points.

Complete the program (begun below) which prints out the correct letter grade for the students. You only need to complete the conditional logic. You DO NOT need to write the code which gets the input from the user.

```
int points; //number of points students earned
boolean ec; //value indicating if student did extra credit


//assume code here which assigns valid values to the variables
```

Quiz 2: If/Else Statements

1. (20 pts) You are driving a little too fast, and a police officer stops you. Write code to compute the result: no ticket, small ticket, or big ticket. If your speed (represented by the integer variable speed) is 60 or less, the result is no ticket. If speed is between 61 and 80 inclusive, the result is a small ticket. If speed is 81 or more, the result is a big ticket. Unless it is your birthday (represented by the boolean variable isBirthday) -- on that day, your speed can be 5 higher in all cases.

Which of the following code segments correctly solve the problem above? Choose ALL that apply.

I.
```
 if(speed <= 60  || (isBirthday && speed <= 65)) {
  System.out.println("No ticket");
} else if(speed <=80 || (isBirthday && speed <= 85)) {
  System.out.println("Small ticket");
} else {
  System.out.println("Big ticket");
}
```

II.
```
if(speed <= 60  && !isBirthday) {
     System.out.println("No ticket");
} else if (speed <= 65) {
     System.out.println("No ticket");
} else if (speed <= 80 && !isBirthday) {
     System.out.println("Small ticket");
} else if (speed <= 85) {
     System.out.println("Small ticket");
} else {
     System.out.println("Big ticket");
}
```

III.
```
if(speed <= 60) {
     System.out.println("No ticket");
} else if (speed <= 65 && isBirthday) {
     System.out.println("No ticket");
} else if (speed <= 80 ) {
     return 1;System.out.println("Small ticket");
} else if (speed <= 85 && isBirthday) {
     return 1;System.out.println("Small ticket");
} else {
     System.out.println("Big ticket");
}
```

2. (30 pts) Assume the statements below are part of a Java program which compiles and runs. What is the output if the user types 20?

```
Scanner in = new Scanner(System.in);
int num = in.nextInt();


System.out.println("one")
if(num <= 40) {
     System.out.println("hi");
} else if (num <= 20) {
     System.out.println("cry");
} else if (num <= 10) {
     System.out.println("bye");
} else {
     System.out.println("sigh");
}


System.out.println("two");
if (num > 30) {
     System.out.println("hi");
} else {
     System.out.println("cry");
} if (num > 20) {
     System.out.println("bye");
} else {
     System.out.println("sigh");
}


System.out.println("three");
if (num % 10 == 0) {
     System.out.println("hi");
} if (num % 20 == 0) {
     System.out.println("cry");
}
if (num % 4 == 5) {
     System.out.println("bye");
} else {
     System.out.println("try");
}
```

Quiz 3: While Loops

1. (25pts) Consider the following pieces of code. Which of the following pieces result in a loop which terminates? That is, which of the following are NOT infinite loops?

I.
```
int x=20;
while (x!=15){
     System.out.println("The value of x is: " + x);
     x /= 4;
}
```

II.
```
int x = 20;
while (x!=5){
     System.out.println("The value of x is: " + x);
     x --;
}
```

III.
```
int x = 20;
while (x!=5){
     System.out.println("The value of x is: " + x);
     x -= 2;
}
```

IV.
```
int x = 20;
while (x!=5){
     System.out.println("The value of x is: " + x);
     x ++;
}
```

V.
```
int x = 20;
while (x!=5){
     System.out.println("The value of x is: " + x);
     x = x-5;
}
```

2. Consider the following code:

```
import java.util.Scanner;
public class Test  {
   public static void main(String[] args) {
      Scanner input = new Scanner(System.in);
      int n, m;
      n = input.nextInt();
      m = n;
      while (n != 0) {
          n = input.nextInt();
          if (n > m){
            m = n;
          }
      }
      System.out.println("m =  " + m);
      System.out.println("n =  " + n);
   }
}
```

Suppose the user types the numbers:

2 3 4 5 0 6

in sequence, hitting [Enter] between each number.

A. (20 pts) What is the output of the above code?

B. (5 pts) Write a one sentence comment to describe the function of this code.

Quiz 4: For Loops

1. (20 pts) Which of the following code segments will print out the values from 5 to 100 (inclusive), counting by 5's? (i.e. 5, 10, 15, … 100) Choose ALL that apply.

I.
```
for(i=100; i>0; i--){
     S.O.P (i%5);
}
```

II.
```
for(i=1; i<=20; i++){
     S.O.P (i*5);
}
```

III.
```
for( i=5; i<=100; i+=5){
     S.O.P ( i);
}
```

IV.
```
for( i=100; i>0; i--){
     S.O.P (i);
}
```

V.
```
for( i=5; i<=100; i-=5){
     S.O.P (i);
}
```

2. Consider the following code segment:

```
String s = "mystery";

for( int i = s.length()-1; i>=0; i--){
     S.O.P (s.charAt(i));
}
S.O.P("----");
for( i=0; i< s.length(); i+=2){
     S.O.P (s.charAt(i));
}
```

A. (20 pts) What is the output of the following code?

B. (10 pts) Write a one sentence comment to describe the function of this code.

Quiz 5: Method Parameters

1. Consider the following method header:

```
public static void first(String second, int third) {…}
```

A. What is the name of this method?

B. How many parameters does this method have?

C. Which of the following are valid ways of invoking or calling the above method? Select all that are valid.

I. `first("second", "third");`

II. `first("Hello", 45);`

III. `first("2nd", 3);`

IV. `first(2, 3);`

V.
```
String s = "s";
int x = 6;
first(s, x);
```

2. Consider the following method:

```
public static void sports(String basketball, String football, String
baseball) {
      System.out.println(baseball + " is better than " + basketball);
      System.out.println("but " + football + " is my favorite sport.");
}
```

A. (18 points) What will be the output of the program if the method above is invoked with the following code?

```
String football = "basketball";
String basketball = "baseball";
String baseball = "football";

sports(baseball, football, basketball);
```

B. (18 points) Using the variables declared and initialized in part (a), write the invocation of the method sports which would generate the following output:

Basketball is better than baseball
but football is my favorite sport.

Quiz 6: Method Return Values

1. Consider the following function:

```
public int changer(String x, int y) {
 x = x + "castle";
 y = y * 3;
 return y + 6;
}
```

The function is called via the following code:

```
String s = "sand";
int n = 5;
n = changer(s, n);
```

A. (2 points) What **type** of value will the function changer return?

B. (2 points) What does it mean if a function has a return type of void?

C. (10 points) What values are assigned to variables s and n when the above code finishes?

2. (36 points) Assume the following code compiles and runs as written. What is the output when the code is run?

```
public static String method1(int two, String three) {
     if(two < 4){
          System.out.println("one");
          return three;
     }
     else{
          return(method2(three));
          System.out.println("three");
     }
}
public static String method2(String two) {
     return "two";
     System.out.println(two);
}
public static void main(String[] args) {
     String one = "two";
     String two = "four";
     String four = "one";
     System.out.println(one);
     System.out.println("four");
     System.out.println(method1(3,four));
     System.out.println(method1(4,two));
}
```

Appendix D: Final Exam Questions

The following questions do not appear in the online thesis repository so that they may be used for future final exams in this course.

Appendix E: Departmental Survey

Department of Mathematics and Computer Science

Course Evaluation Form

Please comment on the strengths and weaknesses of the course:

Please comment on the strengths and weaknesses of the instructor:

Would you recommend this course or instructor to another student? Why or why not?