Teaching Introductory Programming Concepts:

A Comparison of Scratch and Arduino

A Thesis

Presented to

the Faculty of California Polytechnic State University

San Luis Obispo

In Partial Fulfillment

of the Requirements for the Degree

Master of Science in Computer Science

By

Anne Beug

June 2012

© 2012
Anne Beug
ALL RIGHTS RESERVED

COMMITTEE MEMBERSHIP

TITLE: Teaching Introductory
Programming Concepts: A
Comparison of Scratch and Arduino

AUTHOR: Anne Beug

DATE SUBMITTED: June 2012

COMMITTEE CHAIR: Dr. Phil Nico

COMMITTEE MEMBER: Dr. Alex Dekhtyar

COMMITTEE MEMBER: Dr. Chris Lupo

Abstract

Teaching Introductory Programming Concepts: A Comparison of Scratch and Arduino

by Anne Beug

Computing has become an integral part of modern America. The Bureau of Labor Statistics estimates that between 2008 and 2018, the United States will have 1.4 million job openings in computing fields [1]. Since the early 2000's (just after the "dot com bust"), the number of students enrolled in computing majors (Computer Science, Computer Engineering, etc.) dropped precipitously by over 50%, only starting to rise again in 2009 [2]. One way to rectify to this gap between demand and supply is to start training students in computational thinking during high school. While the Computer Science Advanced Placement AB test has been retired, a new AP test called "CS Principles" has been added to the suite of tests [3], highlighting the importance of computing to a modern education. Not only will we need more computing professionals, people in other professions will need to have computing skills. We propose in this thesis a comparison of two computing platforms. Scratch is a well-proven platform that teaches core programming concepts through a graphical programming interface. Arduino boards are open-source microcontrollers with an accompanying development environment and C-like language. We develop a parallel curriculum in Scratch and Arduino and compare the two. While we are unable to draw conclusive results from our quantitative study, from our qualitative research we see that Arduino is unsuitable for teaching core programming concepts to computing novices.

Acknowledgements

I would like to thank Jan Fetcho of San Luis Obispo High School and Donald Repucci of Morro Bay High School for allowing me to use their classrooms for my thesis work. Their guidance and advice were invaluable, and I would not have completed my Master's degree without their work.

And of course, I would like to thank my committee for all their advice and support.

Table of Contents

List of Figures.....	viii
List of Tables.....	ix
1. Introduction	1
1.1. Scratch.....	4
1.2. Arduino.....	7
2. Background and Related Work	10
2.1. Curricula.....	12
2.2. Educational Platforms	14
2.2.1. Scratch	14
2.2.2. Arduino.....	16
3. Methodology.....	17
3.1. Platform Choice.....	17
3.2. Participants.....	17
3.2.1. San Luis Obispo High School.....	18
3.2.2. Morro Bay High School.....	19
3.2.3. Groups.....	20
3.3. Short Lectures	22
3.4. Labs	24
3.4.1. Scratch	25
3.4.2. Arduino	26
3.4.3. Combined Scratch and Arduino.....	29
3.5. Survey.....	31
3.6. Challenges	32
4. Results	34
4.1. Participation	34
4.2. Computing Experience and Attitudes	34
4.2.1. Question 6: Comfort with computers	34
4.2.2. Questions 7 - 8: Previous programming experience.....	35
4.2.3. Question 9: Computing major	37
4.3. Objective Questions	38
4.3.1. Question 1a: Variables.....	39
4.3.2. Question 1b: Variables	40
4.3.3. Question 2: Variables in sequence.....	41

4.3.4.	Question 3: Conditionals	42
4.3.5.	Question 4: Iteration	43
4.3.6.	Question 5: Functions	44
4.4.	Self Assessments	44
4.4.1.	Question 10: Understanding Variables	45
4.4.2.	Question 11: Understanding Conditionals	45
4.4.3.	Question 12: Understanding Iteration	46
4.4.4.	Question 13: Understanding Functions	46
4.4.5.	Question 14: Understanding Programming	47
4.5.	Lab Completion	47
4.6.	Study Experience	48
4.6.1.	Question 15 - 18: Easiness and Interestingness of Platforms	48
4.6.2.	Questions 19 - 20: Image of computing	48
4.6.3.	Question 31: Likes	49
4.6.4.	Question 32: Dislikes	50
5.	Conclusion	52
6.	Future Work	55
	Bibliography	56
	Appendix A: Lectures	60
	Appendix B: Scratch Labs	71
	Appendix C: Arduino Labs	82
	Appendix D: Survey	94
	Appendix E: Detailed Results	98
	Appendix F: Computing Careers Supplemental Survey	141

List of Figures

Figure 1-1: Scratch IDE with simple script.	5
Figure 1-2: Different Scratch block types.	6
Figure 1-3: Arduino IDE with a simple sketch to blink an LED.	8
Figure 1-4: Arduino microcontroller with breadboard and LEDs.	9
Figure 3-1: Breadboard transportation.	27

List of Tables

Table 3-1: Participant groups.	22
Table 3-2: Topics covered in each lecture.....	23
Table 4-1: Question 6 results summary – comfort with computing.....	34
Table 4-2: Question 8 detailed results – previous programming platforms	36
Table 4-3: Programming platform experience for all groups	37
Table 4-4: Question 9 results summary – likelihood to choose computing major	38
Table 4-5: Questions 1 - 5 summary – objective questions	39
Table 4-6: Question 1a results summary – variables.....	39
Table 4-7: Question 1b results summary – variables	40
Table 4-8: Question 2 results summary – variables	41
Table 4-9: Question 3 results summary – conditionals	42
Table 4-10: Question 4 results summary – iteration	43
Table 4-11: Question 5 results summary – functions	44
Table 4-12: Question 10 results summary – understanding variables.....	45
Table 4-13: Question 11 results summary – understanding conditionals.....	45
Table 4-14: Question 12 results summary – understanding iteration.....	46
Table 4-15: Question 13 results summary – understanding functions	46
Table 4-16: Question 14 results summary – understanding computer programming	47
Table 4-17: Summary of completion for Arduino labs	47
Table 4-18: Summary of completion for Scratch labs.....	48
Table 4-19: Question 15 - 18 results summary – easiness and interestingness of platforms.....	48
Table 4-20: Question 19 - 20 summary – change in image of computing	49
Table 4-21: Question 31 – likes.....	50
Table 4-22: Question 32 – dislikes.....	51

1. Introduction

Computing has become the framework of modern daily life. From personal computers to telephones to cars and cameras, much of what we use for work and play, communication and entertainment is controlled by software. To find jobs, housing or love, we turn to the web for information and resources. As foretold in the Spring 2000 Bureau of Labor Statistics Occupational Outlook Quarterly [4], "Employment of computer systems analysts, engineers and scientists is expected to double between 1998 and 2008 as organizations become increasingly dependent on advancing technologies and the workers who design and implement them."

Within a few generations, computers have come to underlie the foundations of all industries from natural sciences to health care to finance and engineering. International commerce, banking, airline flights and local subway schedules are now created, recorded and analyzed via software. The field of biology has completely changed with the advent of software and hardware to process genome data. As many other fields become dependent on computing, professionals in those fields must not only acquire expertise in their own fields, but competency in computing as well. In the globally connected world of the new millennium, software technology serves as a driving force behind the innovation that propels the international economy forward. Fields as diverse as social sciences, economics, computational finance and marketing rely on the digital computing power of software to collect, analyze and transform data sets far too large for humans to manage.

A goal is to move the current generation from being users of technology to creators of technology. To do this, we must teach this generation computing.

Computing offers well-paying careers in positions as varied as software developer, systems analyst, database architect and web developer—all of which were listed in the top ten of US News & World Report's Best Jobs in 2012 [5], with median annual salaries between \$US70,000 and \$US90,000 [6]. Current statistics from the US Bureau of Labor and Statistics rate computing fields among the fastest growing with the number of positions for computing professionals growing 30% from 2010 (913,100 positions) to 2020 (1,184,000 positions) [7]. The 2020 projected workforce (16 years of age and older) for all occupations is 164,360,000 [8]. Software developers will be approximately 0.72% of the total workforce, making more than one in every 140 people working in the US a software developer. And while the number of bachelor's degrees in computing-related majors has seen a rise in the last few years after a steep decline earlier in the decade, there is still a large anticipated gap between open computing positions and computing professionals in the US [9].

While many colleges and universities have computer science programs (258 ABET accredited programs in the US [10]), many high schools do not (less than 40% of US high schools offer AP Computer Science [11] or programming [12]). A 2010 report [13] from the Computer Science Teachers Association (CSTA) and the nation professional society Association for Computing Machinery (ACM), reports the disparity between the growing importance of computing to society and the place computing has in secondary education. In order to feed the pipeline of college graduates with computing degrees or training, this

problem must be addressed before students enter university. Students need to be aware of computing as a skill, major and career. As described in the Curricula Related Work subsection (Section 2.1), we discuss the current state of secondary and post-secondary computing curricula and see that national standards are lacking.

In this study, WE compared two separate software development platforms for teaching introductory core programming concepts to secondary school students, Scratch and Arduino. We developed a full five-session course with lectures and hands on labs in both platforms. The covered concepts are core programming concepts – variables, conditionals, iteration and functions. A complete pilot of the courses was undertaken at two local high schools with a total of 119 students in five separate classes over the course of three months.

We assessed the students' grasp of the programming concepts through a pre-survey and post-survey and compared the difference between the two. Both surveys also ask the students about their computing background and attitudes toward computing as a major, as well as assessment questions about programming concepts.

Changes to the course materials were made during the pilot session as WE discovered incorrect assumptions on my part and observed the students encountering challenges. It was important to not risk the students' education for the sake of the thesis, and we worked closely with their teachers to ensure time spent in the study was worthwhile for the class. Changes are described in the Methodology (Section 2) and Results (Section 4) sections.

The remainder of this section gives an introduction to the two platforms – Scratch (Section 1.1) and Arduino (Section 1.2). In the Background and Related Work (Section

2), we cover relevant research on computing education, Scratch and Arduino. The methodology of this thesis is covered in Methodology (Section 3). Results (Section 4) are presented in summary, and full details of results can be found in Appendix E. Text and slides from lectures, labs and surveys is listed in Appendices A – D and F.

1.1. Scratch

Scratch grew out of MIT's Lifelong Kindergarten Lab, officially launching in 2007 as a new educational programming and computing platform. From the Scratch website [14],

Scratch is designed to help young people (ages 8 and up) develop 21st century learning skills. As they create Scratch projects, young people learn important mathematical and computational ideas, while also gaining a deeper understanding of the process of design. With Scratch, kids can create their own interactive stories, games, music, and animation and share their creations with one another on the Web.

Scratch's visual programming interface allows users to build programs by selecting "blocks" (programming instructions) from a palette on to a script area (as seen in Figure 1-1). The blocks click together like puzzle pieces, preventing syntactical errors.

Figure 1-1: Scratch IDE with simple script.

Scratch incorporates multimedia in the form of 2-dimensional sprites, animation and sound. Sprites are placed on the stage (upper-right-hand area of the IDE). Figure 1-1 shows a screenshot of the Scratch IDE with the default project. Scripts can act on either the sprites or on the stage itself. Script block categories include motion, appearance, sound, drawing, control, sensing, arithmetic operators and variables. Examples of block types are show in Figure 1-2.

<i>Trigger (hat) block</i> – beginning of execution of a stack of command blocks.
<i>Reporter block</i> – returns the current value of a variable.
<i>Loop block</i> – repeats commands enclosed within.
<i>Conditional block</i> – if / else statement executes commands in one section or the other based on given predicate.
<i>Predicate block</i> – returns true or false.
<i>Command block</i> – performs some action.
<i>Operator block</i> – perform math and text operations

Figure 1-2: Different Scratch block types.

Trigger blocks allow for starting scripts based on a given event – the green flag being pressed, a specified key being pressed, or a message broadcast from another sprite. Command blocks perform some functionality, such as moving or changing the color of a sprite. These blocks click together to form a sequence of commands called a "stack." Reporter blocks return a value from either an expression or a variable. Predicate blocks are like reporter blocks, but only return true or false from the evaluation of a conditional expression. Each block is colored to match its category, and the shapes of blocks suggest how to use them. Trigger blocks, which initiate execution of a script, have a curved top that does not interlock with other blocks. Loop blocks are C-shaped and fit around other control blocks. Other control blocks have slots at the top and bottom to fit together with other control blocks. Both predicate and reporter blocks are shaped to fit in to inputs to other blocks.

Scratch has been well researched in academic environments. Other research has studied the effectiveness of Scratch-based curricula in teaching core programming concepts. Other research looks at how Scratch affects students' attitudes toward computing. We cover this research in depth in the Background and Related Work section (Section 2) below.

1.2. Arduino

The Arduino platform consists of a set of microcontrollers, a programming language and an IDE. All components of the platform are open source. The language is based on the Wiring and Processing [15] languages, which were created to teach core programming and computing concepts through electronics and visual arts respectively to non-programmers. The Arduino language is syntactically similar to C and Java.

Figure 1-3: Arduino IDE with a simple sketch to blink an LED.

The Arduino microcontroller designs are also open source [16]. The boards themselves are inexpensive – about \$US27 for the Arduino Uno R3 that WE used in this thesis. We also used various electronic parts – LEDs, rotary potentiometers, momentary pushbuttons, Piezo buzzers and OneWire¹ temperature controllers. Schematics for lab setups can be found in Appendix B.

¹ <http://arduino.cc/playground/Learning/OneWire>

Figure 1-4: Arduino microcontroller with breadboard and Piezo buzzer

The platform was designed to be accessible to novice programmers and electrical engineers, "intended for artists, designers, hobbyists, and anyone interested in creating interactive objects or environments [17]." The platform has, however, recently begun making inroads in to computing education, though primarily in embedded systems courses rather than introductory programming courses [18]. We will discuss uses of Arduinos, few as they are, in education settings in the Background and Related Work (Section 2) section.

2. Background and Related Work

There is both a diverse and large body of research in the area of computer science education, focused on primary, secondary and tertiary schools, starting in the 1970's [19]. One common theme in many papers is the fact that learning and teaching programming is difficult [19-21].

Significant computing education conferences include SIGCSE Technical Symposium on Computer Science Education (SIGCSE) [22], International Computing Education Research Workshop (ICER) [23], Innovation and Technology in Computer Science Education (ITiCSE) [24], Australasian Computing Education Conference (ACE) [25], Koli Calling [26] and CITRENTZ Conference [27]. These conferences focus on a large range of topics from curriculum development, introductory languages, educational programming platforms to common challenges faced by students of computer science. Surveys of papers from these conferences can be found in Sheard et al. [28]. Papers were grouped into areas including ability, aptitude and understanding; teaching, learning and assessment techniques; and teaching, learning and assessment tools.

Robins, Rountree and Rountree provide another survey of psychological and education research of programming [19]. They find a number of distinctions presented in the various papers: experts vs. novices, programming knowledge vs. programming strategies, code comprehension vs. code generation, and the debate between teaching procedural versus objected-oriented programming paradigms.

Papers distinguish the way novices program and the way in which experts program. Novices are dealing with smaller units of programming – comprehending programs line-by-line, whereas experts see programs in larger "chunks" of functionality [29]. Where a novice would see a *for* loop, variable assignment and incrementation of a variable, an expert would see a chunk of code to find the average of a list of numbers. Seeing code line-by-line is referred to as "programming knowledge" and seeing chunks of code as "programming strategy," "plans," or "schemas" [19,28]. These strategies are all blueprints experts have to apply to new problem cases. De Raadt names multiple strategies that experts use: average plan, triangular plan, counter-controlled loop plan, search algorithm, etc. [30]. Novices, however, have a hard time seeing the bigger picture in programming, understanding concepts at a lower level of abstraction [19,29,31,32]. Some authors recommend teaching programming strategies explicitly [28,30,33]. Papers also distinguish between the ability to read code (comprehension) and the ability to write code (generation) [19], and studies find little correspondence between the two abilities [19,32].

Another area of study is specific problems novice programmers encounter. Pea writes about different "conceptual bugs" in novice programming [21]. Novices often attribute intentionality to the machine – they believe the computer somehow knows what they intend for the program to do without explicating such programming. They also see more meaning in their code than what they actually include. One example of this is missing semi-colons (in languages that require them). The novice expects the compiler to find these omissions and understand what she intended. After more than half a semester of Pascal, over 25% of the students still used the same variable for different purposes,

expecting the machine to interpret their intentions. Students also did not understand that their programs were executed sequentially, thinking that multiple things could be happening at the same time.

Simon describes the problems novice programming students had with a simple swap [34]. He notes that assignment in programming conflicts with what students have been taught for years in mathematics. Students do not understand that assignment must happen in order – first variables must be assigned values before they can be used in other statements.

2.1. Curricula

ACM, IEEE and the CSTA (Computer Science Teachers Association) have joined together to develop curricula for computer science education at the primary, secondary and tertiary level.

Computing Curricula 2001 (often referred to as CC2001), and its interim revision Computing Curricula 2008, lay out full curricula for undergraduate studies in computer science. These include studies in programming fundamentals, discrete structures, algorithms and complexity, architecture and organization, operating systems, net-centric computing, programming languages, human-computer interaction, graphics and visual computing, intelligent systems, information management, social and professional issues, software engineering and computational science. Individual curricula exist for computer engineering, software engineering, information systems and information technology. A full revision is under works for 2013 that includes interdisciplinary work and the new

programs in the form of "Computational X." The first ACM computing curriculum was published in 1968 [35].

At the primary and secondary level in the United States is a curriculum for computing for kindergarten through 12th grade. CSTA K-12 [39] provides a "model curriculum" for teaching computer science at different grade levels. It includes not only programming, but also course work in hardware, networks and many other areas similar to that in CC2001. Fundamental computing concepts are introduced in elementary school and integrated with other curricula (math, science, etc.). As with the undergraduate curricula, an emphasis is placed on interdisciplinary work. The primary and secondary school curricula also outline challenges to implementation – the fractured nature of implementation at the local and state level; providing appropriate curriculum materials; adoption in school; and training qualified computer science teachers. The CSTA reports in multiple publications [39,36,11,37] that there is no nation-wide adoption of a common computing curriculum. Few states require computing courses for graduation and, where required, are more often skills-based classes in computer usage, rather than computer science courses.

Another significant curriculum at the high school level is the new AP CS Principles course. This course has been piloted at several high schools and universities [38,39,40]. It explores seven "Big Ideas" in computing that focus on the creative aspects of computer science and, again, the interdisciplinary nature of the field.

2.2. Educational Platforms

Other computing education research has been performed on the Scratch and the Arduino platforms. Caitlin Kelleher and Randy Pausch provide a comprehensive survey of novice programming environments, including Scratch but not Arduino [41].

2.2.1. Scratch

Scratch was originally created as an educational platform, based on the constructionist ideas behind Logo [42,43], where students create knowledge themselves, rather than just absorbing it from teachers. Maloney, et al. [42] give a detailed summary of the Scratch environment. The "wide walls, high ceiling and low floor" principle of making an environment easy to approach, of being capable of allowing diverse projects and of accepting projects with increasing complexity is described by Resnick, et al. [44].

Studies have found various results using Scratch to teach core computing concepts. Colleen Lewis reports on a study comparing Scratch and Logo for teaching programming to 6th graders [45]. Students spent a total of 36 hours over 12 days learning programming constructs shared by Scratch and Logo. She studied both their learning outcomes and attitudes toward computing, but found little difference of statistical significance between the two platforms. One area where students using Scratch outperformed students in the Logo group was in comprehending conditional statements. Surprisingly, however, students in the Scratch group did not show greater comprehension of loops.

In "Habits of Programming in Scratch," Meerbaum-Salant, et al. [46] found that while Scratch encourages self-directed learning, students only really learned programming

concepts when explicitly taught them. The habits they found in novice programmers learning with Scratch include bottom-up programming and extremely fine-grained programming. The students took the bottom-up approach to the extreme, starting with the small components to build up to top-level system and not approaching the system from an algorithmic standpoint. With extremely fine-grained programming, the students took functional decomposition to the extreme as well, breaking scripts up more than necessary and accidentally introducing concurrency bugs.

Rivzi, et al. [47] describe a new Scratch-based undergraduate course (CS0) inserted before the traditional first programming course (CS1) and aimed at student retention. One set of students enrolled directly in CS1, while the others enrolled first in CS0. Amongst the CS0 set, the researchers found increased interest in computer science as well as improved learning outcomes.

A study of Scratch use by younger students (ages 8 – 18) at a Computer Clubhouse is described by Maloney et al. [48]. The researchers studied over 530 projects created in Scratch by youth at the center. Rather than explicitly teaching participants programming concepts, researchers allowed the youth to work on projects of their own choosing. A significant number of projects used loops (51.8%) or conditional statements (26.1%), but fewer used variables (9.6%).

Another study investigated learning results of computer science concepts by students learning in the Scratch environment [49]. A Scratch-based curriculum was developed for middle-school-aged children. Middle school teachers taught the course during regular school hours. Pre, interim and post tests were used to test students' comprehension of

concepts including loops, variables and lists. Students performed well with loop concepts, but less so with variables.

2.2.2. Arduino

Far less research has been performed with Arduino, perhaps because it was not conceived of as an educational platform. Most work describes integrating Arduino into existing microcontroller or robotics courses [50,51,52].

3. Methodology

3.1. Platform Choice

We chose Arduino as our experimental platform to study because of the dearth of research on using Arduino to teach introductory programming education. Arduino has become popular amongst electronics hobbyists [16] and has recently been introduced into embedded systems curricula [16,52].

Scratch, on the other hand, grew out of academic work. Multiple studies have been performed investigating the effectiveness of Scratch in teaching introductory programming concepts [44,46,47,49,53,54]. It works well for some concepts (loops and conditionals) but not for others (variables and functions) [49]. Scratch has still been accepted as a platform for teaching novice programmers in junior high [21], high school and at the university level [44,47,49,53–55].

The other platform we considered for our control group was Alice [56]. However, we chose to use Scratch as it is already used in many of our university outreach programs as well as in the local elementary and secondary school district. Using BYOB in the final lab alleviated our main concern with Scratch – that it does not have the capability to express functions.

3.2. Participants

The participants for our study were from two local high schools – San Luis Obispo High School and Morro Bay High School, both part of California’s Coastal Unified School

District. In the local school district, computer programming classes are considered elective and technical courses rather than college preparatory classes (such as science, math and history). However, the computer applications course is a graduation requirement.

3.2.1. San Luis Obispo High School

San Luis Obispo High is the high school in the same town as Cal Poly. There are 1,530 students in grades 9 through 12, with a student-to-teacher ratio of 22:1 [57]. We worked with one teacher at the school who has a Master's degree in Computing Education from Cal Poly. She teaches four programming classes (including AP Computer Science), computer applications (Word, Excel, Access, etc.) and a computer repair course. With her recommendation, we taught Scratch in her 1st and 3rd period computer applications classes and Arduino to her 2nd period combined Programming I, II and III class. Some of the students in the programming classes have taken previous programming courses, including AP Computer Science (which is currently taught in Java). Programming I and II are taught in Visual Basic and Programming III is an independent projects class. Some students have also taken the computer repair course, in which they perform electronics projects, using resistors, LEDs, potentiometers and breadboards. These projects, however, do not have a software component. The computer programming and repair classes fall under the Regional Occupational Program (ROP) [58].

Each year, the California Department of Education produces a report on schools' performance on standardized tests, called "Adequate Yearly Progress". The score is on a scale of 200 – 1000 points. For the report based on the Spring 2011 testing cycle of the

2010 – 2011 school year, San Luis High scored 827 and had a graduation rate of 97.73% [59].

Overall, we had 74 participants from San Luis High -- 56 from two computer applications classes and 18 from the combined programming classes. The classes were back-to-back during 1st, 2nd and 3rd periods from the start of school until lunchtime. We held one session a week, every Monday, with a two-week hiatus between the first session and the second session for Spring Break.

3.2.2. Morro Bay High School

Morro Bay High is the smaller of the two high schools in the local school district. There are 857 students, with a student-to-teacher ratio of 20:1 [60]. At Morro Bay High, as well as San Luis Obispo High, the computer programming and repair classes fall under ROP. The school offers three programming classes – Programming I, II and III, but not AP Computer Science. There is also a computer repair class where the students learn some basic electronics. The programming classes are taught by the industrial arts (welding, woodworking, drafting, electronics) teacher. He teaches computer applications alongside a second computer applications teacher.

For the 2011 Adequate Yearly Progress report, Morro Bay High scored 807 and had a graduation rate of 95.57% [61].

From Morro Bay High, we had a total of 45 participants – 15 from programming classes and 30 from manufacturing concepts classes. Morro Bay High operates on a block schedule – students are in class for approximately two hours for each class and classes

meet alternate days. At Morro Bay High, we had anywhere from 1.5 to 2 hours per session, depending on what other work students had to do for that day. Because of the way the Morro Bay High schedule is structured and our own scheduling constraints, we sometimes held more than one section per week.

3.2.3. Groups

We worked with a total of five classes, which we have labeled Group 1 – 5. The first and last groups were at Morro Bay High School, and the middle three groups were at San Luis High School. The first and second courses were separated by a few weeks. The second, third and fourth groups were all taught on the same days. The fifth course started two days after the fourth course ended. Details about each group can be found in Table 3-1.

Group 1 (Scratch)	<p><i>School:</i> Morro Bay High</p> <p><i>Participants:</i> 15</p> <table> <tr> <td>Freshman</td><td>40.00%</td></tr> <tr> <td>Sophomores</td><td>11.11%</td></tr> <tr> <td>Juniors</td><td>16.67%</td></tr> <tr> <td>Seniors</td><td>22.22%</td></tr> </table> <p><i>Class:</i> Programming I, II, III</p> <table> <tr> <td>Programming I</td><td>66.67%</td></tr> <tr> <td>Programming II</td><td>13.33%</td></tr> <tr> <td>Programming III</td><td>20.00%</td></tr> </table> <p><i>Session dates:</i></p> <p>Wednesday, February 15, 2012</p> <p>Friday, February 17, 2012</p> <p>Friday, February 24, 2012</p> <p>Tuesday, February 28, 2012</p> <p>Wednesday, March 7, 2012</p>	Freshman	40.00%	Sophomores	11.11%	Juniors	16.67%	Seniors	22.22%	Programming I	66.67%	Programming II	13.33%	Programming III	20.00%
Freshman	40.00%														
Sophomores	11.11%														
Juniors	16.67%														
Seniors	22.22%														
Programming I	66.67%														
Programming II	13.33%														
Programming III	20.00%														

Group 2 (Scratch)	<p><i>School:</i> San Luis Obispo High</p> <p><i>Participants:</i> 29</p> <table> <tr> <td>Freshman</td><td>72.41%</td></tr> <tr> <td>Sophomores</td><td>0.00%</td></tr> <tr> <td>Juniors</td><td>6.90%</td></tr> <tr> <td>Seniors</td><td>0.00%</td></tr> </table> <p><i>Class:</i> Computer Applications</p> <p><i>Session dates:</i></p> <p>Monday, March 26, 2012</p> <p>Monday, April 16, 2012</p> <p>Monday, April 23, 2012</p> <p>Monday, April 30, 2012</p> <p>Monday, May 7, 2012 (40 minute schedule)</p>	Freshman	72.41%	Sophomores	0.00%	Juniors	6.90%	Seniors	0.00%										
Freshman	72.41%																		
Sophomores	0.00%																		
Juniors	6.90%																		
Seniors	0.00%																		
Group 3 (Arduino)	<p><i>School:</i> San Luis Obispo High</p> <p><i>Participants:</i> 18</p> <table> <tr> <td>Freshman</td><td>11.11%</td></tr> <tr> <td>Sophomores</td><td>33.33%</td></tr> <tr> <td>Juniors</td><td>27.78%</td></tr> <tr> <td>Seniors</td><td>22.22%</td></tr> <tr> <td>Unknown</td><td>5.56%</td></tr> </table> <p><i>Class:</i> Programming II, III, AP Computer Science</p> <table> <tr> <td>Programming II</td><td>50.00%</td></tr> <tr> <td>Programming III</td><td>11.11%</td></tr> <tr> <td>AP Computer Science</td><td>11.11%</td></tr> <tr> <td>Unknown</td><td>27.78%</td></tr> </table> <p><i>Session dates:</i></p> <p>Monday, March 26, 2012</p> <p>Monday, April 16, 2012</p> <p>Monday, April 23, 2012</p> <p>Monday, April 30, 2012</p> <p>Monday, May 7, 2012 (40 minute schedule)</p>	Freshman	11.11%	Sophomores	33.33%	Juniors	27.78%	Seniors	22.22%	Unknown	5.56%	Programming II	50.00%	Programming III	11.11%	AP Computer Science	11.11%	Unknown	27.78%
Freshman	11.11%																		
Sophomores	33.33%																		
Juniors	27.78%																		
Seniors	22.22%																		
Unknown	5.56%																		
Programming II	50.00%																		
Programming III	11.11%																		
AP Computer Science	11.11%																		
Unknown	27.78%																		
Group 4 (Scratch)	<p><i>School:</i> San Luis Obispo High</p> <p><i>Participants:</i> 27</p> <table> <tr> <td>Freshman</td><td>66.67%</td></tr> <tr> <td>Sophomores</td><td>11.11%</td></tr> <tr> <td>Juniors</td><td>3.70%</td></tr> <tr> <td>Seniors</td><td>11.11%</td></tr> <tr> <td>Unknown</td><td>7.41%</td></tr> </table> <p><i>Class:</i> Computer Applications</p> <table> <tr> <td>Computer Applications</td><td>96.30%</td></tr> <tr> <td>Programming III</td><td>3.70%</td></tr> </table> <p><i>Session dates:</i></p> <p>Monday, March 26, 2012</p> <p>Monday, April 16, 2012</p> <p>Monday, April 23, 2012</p> <p>Monday, April 30, 2012</p> <p>Monday, May 7, 2012 (40 minute schedule)</p>	Freshman	66.67%	Sophomores	11.11%	Juniors	3.70%	Seniors	11.11%	Unknown	7.41%	Computer Applications	96.30%	Programming III	3.70%				
Freshman	66.67%																		
Sophomores	11.11%																		
Juniors	3.70%																		
Seniors	11.11%																		
Unknown	7.41%																		
Computer Applications	96.30%																		
Programming III	3.70%																		

Group 5 (Arduino)	<i>School:</i> Morro Bay High	
	<i>Participants:</i> 30	
	Freshman	66.67%
	Sophomores	14.81%
	Juniors	3.70%
	Seniors	11.11%
	Unknown	3.70%
	<i>Class:</i> Manufacturing Concepts	
	<i>Session dates:</i>	
	Wednesday, May 9, 2012	
	Friday, May 11, 2012	
	Tuesday, May 15, 2012	
	Thursday, May 17, 2012	

Table 3-1: Participant groups.

3.3. Short Lectures

Each session started with a short (approximately 10 - 15 minutes) lecture on programming concepts to be covered in the day's lab. Table 2 includes a listing of all topics covered in each lecture and all lecture slides can be found in Appendix A. The same lectures were designed for both labs – Scratch and Arduino.

Session 1 – Introduction	<ul style="list-style-type: none"> • What a computer is • What programming is • Algorithms • Source code • Difference between hardware and software • Arduino basics (only with Arduino group) • Scratch basics (only with Scratch group)
Session 2 – Variables	<ul style="list-style-type: none"> • Variable basics • Variable declaration • Variable assignment • Expressions • Data types • Constants • Collections basics
Session 3 – Conditionals	<ul style="list-style-type: none"> • Conditional basics • Booleans • Compound conditionals

Session 4 – Iteration	<ul style="list-style-type: none"> • Iteration basics • Conditional-controlled loops • Arrays in loops
Session 5 – Functions	<ul style="list-style-type: none"> • Function basics • Parameters • Return values

Table 3-2: Topics covered in each lecture.

The goal of each lecture was to give students enough knowledge to work through the lab for the session. A single lecturer—the author—delivered all lectures.

The Scratch and Arduino lectures were added after the first session at Morro Bay High to give students a better introduction to the platforms and language syntax.

For Groups 1, 2 and 4, we presented examples of each concept in pseudo-code. We wanted all students to have exposure to text-based programming, since this is the most common form of programming. With the examples, we walked the students through the first example, asking them to guess what the code was doing and explaining explicitly concepts like sequential execution and the difference between the assignment operator ("=") and the equality operator ("=="), based on advice from Simon [34]. For the remaining examples, we asked the students to explain what was happening in the pseudo-code, correcting any misinterpretations they had.

Because Group 3 already had significant programming experience (at least one semester), we did not present formal lectures for each concept as we did with the previous groups. We did present a short lecture during the first session on the basics of the Arduino environment. The students were able to browse the language reference on their own to find details for any built-in functions they wished to use.

With Group 5, rather than presenting formal lectures on the given concept for the day, we walked students through creating an example program for each lab. For the first day, the overhead projector was broken, so we walked the students through PowerPoint slides on their own machines. The slides showed the audience the different components of the Scratch programming environment. For Labs 2 through 6, we built a small similar program with the students watching on the overhead projector.

3.4. Labs

All lab write-ups can be found in Appendix B (Scratch) and C (Arduino).

Once the lecture portion was given, the students had the rest of the class period to work on the lab. At San Luis High, this was approximately 30 – 40 minutes, depending on the day and class period. For instance, the daily bulletin was read during the 2nd period programming class (Group 3). On one occasion (Session 2), the bulletin took a full 8 minutes of class time. The two computer application classes also differed in how long they took to complete the labs. For Scratch Sessions 3 and 4, the earlier 1st period class finished the lab in less time than the 3rd period class; however, the later class also kept working in Scratch even after they completed the lab. The class teacher surmised this could be because the 1st period class students were behind on their other class assignments. Morro Bay High has a block schedule in which each class meets every other day for 110 minutes. After various administrative tasks, we had between 50 and 60 minutes for the actual lab work.

While each lab focused on a particular concept, most of the labs required students to use multiple concepts to complete their work. For instance, in order to complete the sound lab (on iteration), students also had to be able to use variables, conditionals and functions.

3.4.1. Scratch

Each Scratch lab was a single project for the student to complete. They were given a project to start from, with existing sprites and scripts to be built on. On occasions when multiple students were having trouble surmounting the same or similar obstacles, we walked through the scripting on the screen in front of the whole class. In Session 2, where the students were to build a MadLibs game, we had to show the students what MadLibs entailed. We mistakenly assumed they were already familiar with the game. For Session 4, the original plan was for students to use Scratch's sound capabilities to create their own song and animation. However, 25 students in a classroom all playing sound became quickly overwhelming, and it was difficult for students to hear their own projects over the din of everyone else's songs. We quickly reworked the lab for Session 4 in the hour between Group 2 and Group 4 to move from creating sound to having the sprites and scripts draw pictures. Both the drawing and the sound appeared to engage a number of students who had previously not expressed interest in the labs – they spent the entire time working on their projects while other classmates moved on to assignments that were due later in the week. For the drawing version of Session 4 (note that we did not update the write-up, just the sample lab we gave the students), we showed the students on the screen how to use Scratch's drawing pen function.

We considered Session 5 (functions) to be the most complex of the Scratch labs. On the day we presented the session, school followed an assembly session, which gave the students only 40 minutes total for the class – 30 minutes to spend on the lab. Based on discussions with the teacher on how to approach the time constraint, we decided to spend more time walking the students through building an example function and then having them use that function as a template to build their own. However, the regular teacher was absent and a substitute was present instead. The class situation was not ideal, and we did not make it through the lab. In addition, it took 15 minutes to get the teacher work station up and running to show to the presentation to the 1st period (Group 2), so that group had even less time.

3.4.2. Arduino

We started Group 1 before the Arduino IDE software had been installed on the school computers. We used small Hewlett Packard laptops for Sessions 1 and 2. By Session 3, the software had been installed, though there were some problems with the device drivers, which we'll explain further in the *Challenges* section below. After the software was installed, the students worked on their own school machines. For the first session, we had transported the electronics (board, breadboard, LEDs, resistors) disassembled and gave them to a few students to assemble at the beginning of class. This caused challenges as some of the boards were put together incorrectly, and the students trying to use them did not have enough electronics background to troubleshoot the boards. We spent much of the period getting the students to the point where they could start programming and running their programs on their Arduinos, but all the students were able to finish the lab with some help. For the remaining sessions, we assembled the breadboards ahead of time

and transported them flat on a cookie sheet, allowing the students to use more of the lab time to program. They did still need to connect wiring from their breadboards to their Arduinos. For the Session 5 lab, which dealt with a temperature sensor, we provided a small test program that would let the students know if their equipment was assembled properly. For all other sessions, the students could tell through LEDs or buzzers whether or not their program was working on their Arduino boards.

Figure 3-1: Breadboard transportation.

Group 3, at San Luis High, also did the Arduino labs. In contrast to Group 1 at Morro Bay High, the students in Group 3 had more programming experience. Many had also taken the computer repair class and worked with electronics before. One student in the class was a previous SkillsUSA Computer Maintenance Technology national champion. Since the students had all had at least one programming course (in Visual Basic or Java), we skipped the concept lectures with this group, only giving them the Arduino lecture. They were all familiar with the concepts of variables, conditionals, iteration and functions.

San Luis High bought 25 Arduinos to use with our research and going forward. The shipment had not arrived in time for Session 1, so we borrowed Arduinos from Morro

Bay High School. Six students had to work in pairs. The software install worked more smoothly at San Luis than at Morro Bay, and the students were able to get started programming and running their programs on the Arduinos right away. The wiring also did not pose a problem for the students. Many of them had already had introductory electronics, and they helped those who did not.

During this first session, the students were able to complete the lab (making an LED blink at different rates) well before the end of the class period. They took the initiative to look up the Arduino API online. After finding the random number generation function, they made their light blink at random intervals.

For Session 2, the Arduinos had arrived, and we used the high school's boards and electronics kits. The students again did the wiring themselves but struggled more with the wiring in Session 2 than in the first session. The potentiometers (variable dial resistors) the students had in their kits were different from the ones displayed in the lab drawing. For the remaining labs, we took the breadboards to the class pre-assembled; the students only had to connect the breadboards to the Arduinos.

We changed the third lab after watching the students at Morro Bay work on it. Instead of the three LEDs simulating a traffic light, we had a push button control and LEDs. All students completed this lab using conditional statements as the lab instructed, though they could have also set the LED to the state of the pushbutton. Many also completed the extra practice session, having the pushbutton control multiple LEDs, with some mirroring the state of the pushbutton and the others displaying the converse.

In the fourth lab, the students looked up songs online and were able to reproduce the Imperial March (from the *Star Wars Episode V: The Empire Strikes Back*, composed by John Williams), among other songs, with their Piezo buzzers. With the first four sessions, the students were able to not only complete the lab, but the extra practice sessions as well.

The final Arduino sensor involved reading temperatures from a small OneWire temperature sensor. The students' task for the lab was to create a function to convert the temperature value returned by the sensor from degrees Celsius to degrees Fahrenheit. This required them to import the sensor libraries into their environment. We gave the students detailed instructions on how to do this, though some still had problems with the IDE recognizing their libraries. This took enough time from the already shortened class period that some of the students were not able to complete the fifth lab.

3.4.3. Combined Scratch and Arduino

Our final group, Group 5, was the Manufacturing Concepts class at Morro Bay High School. The course was presented during four sequential class periods over two weeks. The class covers many different fields, including a section on electronics. The students had completed a section in which they built different electronic circuits in hardware using LEDs, resistors, potentiometers, batteries and speakers. Thus, they were already familiar with breadboards, electronic components and wiring before working with the Arduino boards. The manufacturing class, however, does not have any regular software section of its curriculum.

After discussing different scenarios with the teacher, we decided on an eight-hour course of both Scratch and Arduino. The teacher was instinctively hesitant to start the students out on Arduino but wanted then to have some experience with the boards. For the first two class periods, the class worked on Scratch Labs 2 (Variables – MadLibs), 3 (Conditionals – Tag Game) and 4 (Iteration – Sound). The last two class periods were spent on Arduino Labs 1 (Introduction – Blinking Light), 2 (Variables – Light Dimmer) and 4 (Iteration – Piezo Buzzer).

After the feedback from the computer applications classes at San Luis High (Groups 2 and 4), we made drastic changes to the lecture section of the class. Instead of standing in front of the class lecturing on individual concepts, we showed the class how to accomplish different tasks in Scratch and Arduino. We still gave the students the same labs as Groups 1 – 4 but introduced the concepts inside of Scratch and Arduino. We showed them how to use the different programming constructs and then gave them the rest of the time to work on the labs themselves.

Students also worked collaboratively, asking each other how they accomplished different features of their programs. One student even discovered the Scratch website – he downloaded example projects and incorporated parts into his own.

For the Arduino labs, the students worked in groups of three – the same groups they had for their electronics section. We took the breadboards to the class pre-wired, and the students wired the breadboards to the Arduinos.

The students had also been discussing computer security and information privacy as a class. We spent approximately 30 minutes the last day talking as a group about the

implications of unencrypted pacemaker communication, digital facial recognition and governmental monitoring of digital communications.

3.5. Survey

Our validation mechanism consists of a test and survey given both at the beginning of Session 1 as well as at the end of Session 5. We gave the same test and survey to all groups regardless of whether their labs were in Scratch or Arduino, though we did change the survey slightly twice. After the first group, we inserted Question 2. And after the three San Luis High groups (3, 4, 5), we altered Question 3 to use words ("and", "or", "equals") instead of symbols ("&&", "||", "==").

The survey consisted of four parts. The first section asked basic demographic information including class level in school and what class (programming, computer applications, etc.) they were participating in. The students were asked to choose their own survey identification so that their privacy would be protected.

The next portion of the survey asked the students code comprehension questions to assess their knowledge of the given programming concepts. The questions were short answer (i.e. "what is the value of X?"). We had two design goals for this section: first, to make it objective to grade in such that the student could get the question either correct or incorrect. And second, we needed to use the same test across all Scratch and Arduino groups. Because some of the students would be familiar with text-based code and others only with graphical code, we chose code comprehension questions rather than code generation questions.

The third portion consisted of attitudinal questions (i.e. "do you plan on majoring in computing?").

The final section was only given in the post-survey. It asked students about their experiences with Scratch or Arduino. We discuss the results of the survey in the next section.

A full listing of the survey questions can be found in Appendix D.

3.6. Challenges

One of the primary challenges was the limited amount of classroom time available. We suspected we could have easily filled three times as much classroom time on the covered subjects to give students in-depth knowledge of the concepts. The concepts we covered were not light, and learning to program is not easy [20,21]. As seen in the results section below, many of the students found both Scratch and Arduino "confusing." We covered a lot of material in less than five classroom hours.

The class work was also not required and some did not follow the lab instructions. We did walk around the class to ask students how much of the lab they had completed, which kept them somewhat on track. We found it often necessary to repeat things throughout the class period, as the students did not hear the instructions or were distracted by classmates or their computers. We did find teaching easier the more experience we had with it.

Another significant challenge was particular to Arduino. Dealing with hardware added another layer of complexity to the labs. The students would at times accidentally alter the

wiring on their setup. Though they had wiring diagrams in their labs, most were not able to follow the diagrams. They did not know if their unexpected results were from programming errors or problems with their wirings. We also experienced significant problems with communication between the students' computers and the Arduino boards. Both schools run on Windows XP. At San Luis High, two students needed to reinstall the Arduino drivers each session. At Morro Bay High, half the students had to do the reinstall, and a few had to switch computers (which cost the team 10 minutes) to a working environment.

A major threat to the validity of our experiment is the fact that we performed the Arduino labs with the groups of programming students, many of whom had programming experience, while performing the Scratch sessions with the computer application classes. This was done on request of the class teachers who suspected that Arduino would be overly complex to teach students with no programming experience. The students in the programming class also displayed an existing interest in programming by choosing to take the elective class.

We also were not able to control other variables between the classes. The schedules vary greatly between the two classes. The first group at Morro Bay High (Group 1) was in the afternoon, while all the others were in the morning before lunchtime. Many of the students in Group 1 were absent from one or more class periods to attend sporting events. And in the other groups, students were absent for other varied reasons. The result of this can be seen in the number of students who chose to participate in the study, but were only present for either the pre- or post-survey, but not both.

4. Results

4.1. Participation

In all groups, there were a total of 119 participants. Of those participants, 93 completed the pre-survey (78.15%) and 85 (71.43%) the post-survey; 59 (49.58%) completed both the pre- and post-surveys, 34 (28.57%) only the pre-survey and 26 (21.85%) only the post-survey. In the results section, we use only results from participants who completed both surveys.

4.2. Computing Experience and Attitudes

The survey posed a number of questions to participants regarding their attitude toward computing.

4.2.1. Question 6: Comfort with computers

How comfortable are you with computers? (On a scale from 1 – 5, 1 = not comfortable at all; 5 = very comfortable)		
Group	Mean Δ (Std Dev)	Mean (Std Dev)
1 (Arduino)	Increase 0.50 (0.50)	pre = 4.00 (0.58) post = 4.57 (0.49)
2 (Scratch)	No change 0.00 (0.43)	pre = 3.09 (1.50) post = 3.09 (1.44)
3 (Arduino)	Increase 0.33 (0.62)	pre = 4.00 (0.88) post = 4.42 (0.49)
4 (Scratch)	Decrease -0.50 (0.50)	pre = 3.38 (0.86) post = 2.88 (0.93)
5 (Both)	Decrease -0.21 (0.77)	pre = 3.42 (1.04) post = 3.21 (1.00)
All	Decrease -0.02 (0.69)	pre = 3.54 (1.11) post = 3.56 (1.17)

Table 4-1: Question 6 results summary – comfort with computing

The results show on average participants experienced a slight increase in their comfort with computers, with ten students reporting an increase and 40 students reporting no change. However, a number of students (9 participants, 16%) reported a drop in their comfort level. Two different reasons could be postulated for this. First, expanding their experience with computing could also expand the amount of computing the participants were not comfortable with. And second, the drop in reported comfort level could be within the error range for the question.

4.2.2. Questions 7 - 8: Previous programming experience

Outside of Groups 1 and 4 (the two programming classes), participants reported little to no programming experience. Of those in all groups who did report programming experience in either pre- or post-survey, Visual Basic, Java and Scratch were the most widely reported. Visual Basic is used at both high schools in the Programming I and Programming II courses. Java is used in the AP Computer science class at San Luis High School, which Morro Bay High School does not offer. Scratch is used at local middle schools. Table 4-2 displays computing experience for each individual group, and Table 4-3 shows computing platform experience for all groups as a whole.

Question 7			
How many years programming experience have you had? _____			
Question 8			
If so, with what platforms? C/C++ Java Scratch Visual Basic Arduino Other _____			
	Years Programming		Platform Experience
Group 1 (Arduino)	Mean	0.75	Scratch 42.86%
	Std Dev	0.72	Java 0.00%
	Mode	0	Visual Basic 71.43%
			C/C++ 14.29%
			Other 28.57%
Group 2 (Scratch)	Mean	0.45	Scratch 8.33%
	Std Dev	0.78	Visual Basic 16.67%
	Mode	0	Java 16.67%
			C/C++ 8.33%
			Other 8.33%
Group 3 (Arduino)	Mean	2.12	Scratch 7.69%
	Std Dev	1.51	Visual Basic 92.31%
	Mode	1	Java 38.46%
			C/C++ 15.38%
			Other 23.08%
Group 4 (Scratch)	Mean	0.00	IE 12.50%
	Std Dev	0.00	Word 12.50%
	Mode	0	
Group 5 (Both)	Mean	0.18	Scratch 26.32%
	Std Dev	0.37	Java 26.32%
	Mode	0	Other 26.32%

Table 4-2: Question 8 detailed results – previous programming platforms

Question 8	
Visual Basic	36.00%
Java	18.00%
Scratch	16.00%
C/C++	8.00%
Objective C	2.00%
Arduino	0.00%
Other	20.00%

Table 4-3: Programming platform experience for all groups

4.2.3. Question 9: Computing major

Question 9 (Table 4-4) asks participants how likely they would be to choose a computing major. Of the five groups, only the first displayed an increase in likelihood to major in computing. Group 4 showed no change, and the other three groups showed a decrease. For the increase in Group 1, the change was participants going from being completely unlikely to less unlikely. For Group 3, three participants (23.07%) went from being very likely or likely to choose a computing major to being willing to consider a major in computing—two were seniors and one a sophomore. Eight participants (61.58%) had no change in their likelihood to choose a computing major, and one participant had a one-point increase. Group 3 had the highest average response to Question 9.

How likely would you be to choose a computing major in college? (Computer Science, Computer Engineering, Software Engineering, Information Science, etc.) (On a scale from 1 – 5, 1 = no chance; 3 = would consider it; 5 = definitely will)		
Group	Mean Δ (Std Dev)	Mean (Std Dev)
1 (Arduino)	Increase 0.39 (0.45)	pre = 3.00 (1.51) post = 3.29 (1.16)
2 (Scratch)	Decrease -0.18 (0.39)	pre = 1.91 (1.16) post = 1.73 (1.21)
3 (Arduino)	Decrease -0.25 (0.72)	pre = 4.08 (0.83) post = 3.92 (0.95)
4 (Scratch)	No change 0.00 (0.00)	pre = 1.13 (0.33) post = 1.13 (0.33)
5 (Both)	Decrease -0.16 (0.81)	pre = 2.16 (0.99) post = 2.00 (1.26)
All	Decrease -0.10 (0.63)	pre = 2.50 (1.41) post = 2.39 (1.46)

Table 4-4: Question 9 results summary – likelihood to choose computing major

4.3. Objective Questions

On the whole, we saw little significant change between the students' answers to the objective learning assessment questions in the pre-survey and the post-survey for any of the groups. Of the students with previous computing experience (Groups 1 and 3), most were able to correctly answer the questions in the pre-survey. And of the students with little or no previous programming exposure, few were able to answer Questions 1b, 3 and 4 correctly in the post-survey if they did not answer it correctly in the pre-survey.

	Q1a – Variables	Q1b – Variables	Q2 – Variables	Q3 – Conditionals	Q4 – Iteration	Q5 - Functions
Group 1 (Arduino)	Increase	N/A	N/A	Decrease	Increase	Increase
Group 2 (Scratch)	N/A	No Change	No Change	Increase	Decrease	Increase
Group 3 (Arduino)	N/A	No Change	Increase	Decrease	Decrease	No Change
Group 4 (Scratch)	N/A	Decrease	Increase	Decrease	No Change	Increase
Group 5 (Both)	N/A	No Change	Increase	Decrease	No Change	Increase
All	N/A	Decrease	Increase	Decrease	Increase	Increase

Table 4-5: Questions 1 - 5 summary – objective questions

4.3.1. Question 1a: Variables

<p>For the following questions, assume:</p> <p>$x = 10$ $y = 5$ $z = 1$</p> <p>To what do the following expressions evaluate?</p> <p>a) $x + y$ _____ b) $x - y$ _____</p>		
Group	Change	Aggregates
1 (Arduino)	Increase 0.43 (1.05)	pre = 2.57 (1.05) post = 3.00 (0.00)

Table 4-6: Question 1a results summary – variables

This question was only given to Group 1. A modified version was given to Groups 2 -5.

The increase in the mean score was due to a single participant correctly answering both questions in the post-survey.

4.3.2. Question 1b: Variables

For the following questions, assume: $x = 10$ $y = 5$ $z = 1$ To what do the following expressions evaluate? a) $x + y$ _____ b) $x - y$ _____ c) $z + x + y$ _____		
Group	Mean Δ (Std Dev)	Mean (Std Dev)
2 (Scratch)	No Change 0.00 (0.00)	pre = 2.75 (0.83) post = 2.75 (0.83)
3 (Arduino)	No Change 0.00 (0.00)	pre = 3.00 (0.00) post = 3.00 (0.00)
4 (Scratch)	Decrease -0.38 (0.99)	pre = 3.00 (0.00) post = 2.63 (0.99)
5 (Both)	No Change 0.00 (0.00)	pre = 3.00 (0.00) post = 3.00 (0.00)
2 – 5	Decrease -0.06 (0.41)	pre = 2.94 (0.41) post = 2.88 (0.58)

Table 4-7: Question 1b results summary – variables

This question was given only to Groups 2 – 5. We added *part c* after the initial session with Group 1. We explicitly covered the concepts of variables and expressions with Groups 1, 2, 4 and 5. In each of these groups, we asked students if they were familiar with the concept of variables.

We did not explicitly cover these concepts with Group 3, because of their previous programming experience. Overall, we saw little change in the students' answers between the pre-survey and the post-survey because most students answered all three parts of Question 1b correctly on the pre-survey. The decrease in Group 3 on was due to a single student not answering the question.

4.3.3. Question 2: Variables in sequence

At the end of the following code, what is c equal to? <pre> a = 3 b = a - 1 a = b * 2 c is _____ c = a + b + 1 </pre>		
Group	Mean Δ (Std Dev)	Mean (Std Dev)
2 (Scratch)	No Change 0.00 (0.58)	pre = 0.42 (0.49) post = 0.42 (0.49)
3 (Arduino)	No Change 0.00 (0.00)	pre = 3.00 (0.00) post = 3.00 (0.00)
4 (Scratch)	Increase 0.25 (0.43)	pre = 0.13 (0.33) post = 0.38 (0.48)
5 (Both)	Increase 0.21 (0.61)	pre = 0.42 (0.49) post = 0.71 (0.46)
2 – 5	Increase 0.13 (0.52)	pre = 0.48 (0.50) post = 0.62 (0.49)

Table 4-8: Question 2 results summary – variables

This question was only given to groups 2 – 5. It was added after finding research [34] showing that novice programmers had a difficult time (even after a semester of programming class) recognizing that variable assignment occurs (in procedural and object-oriented languages) in sequence. This concept was explained explicitly to Groups 2, 4 and 5. Group 2 showed no aggregate improvement – one participant answered the question correctly in the pre-survey and incorrectly in the post-survey, and another participant the opposite. Groups 4 and 5 showed increased scores on this question.

4.3.4. Question 3: Conditionals

<p>For the following questions, assume: $x = 10$ $y = 5$ $z = 1$</p> <p>Which of the following conditionals evaluate to true and which to false? (&& means "and", means "or", == means "equals")</p> <p>a) $x > y$ _____ b) $x == y$ _____ c) $(x > 9) \&\& (y > 10)$ _____ d) $(x < 0) (y < 0) (z > 0)$ _____</p>		
Group	Mean Δ (Std Dev)	Mean (Std Dev)
1 (Arduino)	Decrease -0.14 (0.64)	pre = 3.43 (0.49) post = 3.29 (0.45)
2 (Scratch)	Increase 0.58 (1.44)	pre = 2.67 (1.60) post = 3.25 (1.16)
3 (Arduino)	Increase 0.08 (0.36)	pre = 0.85 (0.36) post = 0.92 (0.27)
4 (Scratch)	Decrease -0.38 (1.58)	pre = 2.63 (1.22) post = 2.25 (1.79)
5 (Both)	Decrease -0.05 (1.36)	pre = 3.00 (0.86) post = 2.95 (1.00)
All	Decrease -0.03 (1.33)	pre = 3.15 (1.12) post = 3.12 (1.22)

Table 4-9: Question 3 results summary – conditionals

Most participants in all groups were able to answer Parts A and B of Question 3 correctly. Participants had more problems with Part C and especially with Part D. The question text was changed between Group 4 and Group 5 based on feedback from the high school teacher. The text was updated to use prose ("and," "equals," "or") in Parts A – D instead of symbols ("&&", "||", "==") with an explanation of what the symbol meant. The lectures for Groups 2 and 4 did cover the symbols and meanings. All groups, with the exception of Group 2, showed a decline in score on Question 3.

4.3.5. Question 4: Iteration

What does the following loop output? (Remember, y = 5) <pre> while (y < 12) { print "Hello " y = y + 2 } </pre>		
Group	Mean Δ (Std Dev)	Mean (Std Dev)
1 (Arduino)	Increase 0.43 (0.49)	pre = 0.14 (0.35) post = 0.57 (0.49)
2 (Scratch)	Decrease -0.08 (0.28)	pre = 0.17 (0.37) post = 0.08 (0.28)
3 (Arduino)	Decrease -0.08 (0.62)	pre = 0.69 (0.46) post = 0.62 (0.49)
4 (Scratch)	No Change 0.00 (0.00)	pre = 0.00 (0.00) post = 0.00 (0.00)
5 (Both)	No Change 0.00 (0.00)	pre = 0.00 (0.00) post = 0.00 (0.00)
All	Increase 0.02 (0.39)	pre = 0.20 (0.40) post = 0.22 (0.41)

Table 4-10: Question 4 results summary – iteration

Groups 2, 4 and 5 had low scores on Question 4 in both the pre- and post-surveys – both Group 4 and 5 had no participant with the correct answer in either the pre- or the post-survey.

Group 1 showed an improvement from approximately 15% with the correct answer to almost 60% with the correct answer. Group 3 showed a slight decrease in score – three students who had gotten the answer correct in the pre-survey answered incorrectly in the post-survey. A number of students answered with a variant of the correct output ("Hello Hello Hello "), but with an incorrect number of iterations (three or five "Hellos").

4.3.6. Question 5: Functions

Given the following functions, what does function3 output? <pre>function1() { print "My name " } function2(name) { print "is " name } function3() { call function1(), call function2("Annie") }</pre>		
Group	Mean Δ (Std Dev)	Mean (Std Dev)
1 (Arduino)	Increase 0.05 (0.69)	pre = 0.37 (0.48) post = 0.42 (0.49)
2 (Scratch)	Increase 0.25 (0.60)	pre = 0.33 (0.47) post = 0.58 (0.49)
3 (Arduino)	No Change 0.00 (0.39)	pre = 0.92 (0.27) post = 0.92 (0.27)
4 (Scratch)	Increase 0.25 (0.43)	pre = 0.13 (0.33) post = 0.38 (0.48)
5 (Both)	Increase 0.05 (0.69)	pre = 0.37 (0.48) post = 0.42 (0.49)
All	Increase 0.15 (0.58)	pre = 0.42 (0.49) post = 0.58 (0.49)

Table 4-11: Question 5 results summary – functions

We articulated usage and structure of functions will all but Groups 3 and 5, which had previous experience with functions in programming and showed no change. With Group 5, we did not cover either functions lab. All groups, except Group 3, showed an improvement in their score on the question regarding functions.

4.4. Self Assessments

This section of the survey (Questions 10 – 14) asked students to assess their own understanding of variables, conditionals, iteration, functions and computer programming in general.

In most sections, participants reported an increase in their understanding of the given concepts. There are three exceptions. First, Groups 2, 3 and 4 did not report any change in their understanding of variables. And Group 3 reported a decrease in their understanding of both iteration and programming in general.

4.4.1. Question 10: Understanding Variables

How well do you understand variables? (On a scale from 1 – 5, 1 = not at all; 3 = somewhat; 5 = very well)		
Group	Δ Mean (Std Dev)	Mean (Std Dev)
1 (Arduino)	Increase 1.14 (1.12)	pre = 2.86 (0.83) post = 4.00 (0.76)
2 (Scratch)	No Change 0.00 (0.85)	pre = 2.73 (1.48) post = 2.73 (1.48)
3 (Arduino)	Increase 0.33 (0.47)	pre = 4.54 (0.50) post = 4.92 (0.28)
4 (Scratch)	No Change 0.00 (0.58)	pre = 2.86 (0.83) post = 3.17 (1.07)
5 (Both)	Decrease -0.16 (1.60)	pre = 2.95 (1.76) post = 2.79 (1.28)
All	Increase 0.20 (1.20)	pre = 3.25 (1.48) post = 3.44 (1.41)

Table 4-12: Question 10 results summary – understanding variables

4.4.2. Question 11: Understanding Conditionals

How well do you understand conditionals? (On a scale from 1 – 5, 1 = not at all; 3 = somewhat; 5 = very well)		
Group	Δ Mean (Std Dev)	Mean (Std Dev)
1 (Arduino)	Increase 1.43 (1.4)	pre = 2.86 (0.99) post = 4.00 (0.76)
2 (Scratch)	Increase 1.00 (0.95)	pre = 1.64 (1.23) post = 2.64 (1.37)
3 (Arduino)	Increase 0.15 (1.35)	pre = 3.85 (0.95) post = 4.33 (0.62)
4 (Scratch)	No Change 0.00 (0.58)	pre = 2.00 (0.76) post = 2.67 (1.49)
5 (Both)	Increase 0.74 (0.91)	pre = 1.42 (0.75) post = 2.16 (0.99)
All	Increase 0.79 (1.03)	pre = 2.26 (1.34) post = 3.05 (1.42)

Table 4-13: Question 11 results summary – understanding conditionals

4.4.3. Question 12: Understanding Iteration

How well do you understand iteration? (On a scale from 1 – 5, 1 = not at all; 3 = somewhat; 5 = very well)		
Group	Δ Mean (Std Dev)	Mean (Std Dev)
1 (Arduino)	Increase 1.29 (0.70)	pre = 2.57 (0.90) post = 4.14 (0.83)
2 (Scratch)	Increase 0.82 (1.11)	pre = 1.73 (1.29) post = 2.55 (1.44)
3 (Arduino)	Decrease 0.23 (0.80)	pre = 3.85 (0.77) post = 3.92 (0.64)
4 (Scratch)	Increase 0.00 (0.58)	pre = 1.57 (0.73) post = 2.17 (1.21)
5 (Both)	Increase 1.53 (1.09)	pre = 1.47 (0.94) post = 3.00 (1.30)
All	Increase 0.93 (1.22)	pre = 2.21 (1.35) post = 3.16 (1.33)

Table 4-14: Question 12 results summary – understanding iteration

4.4.4. Question 13: Understanding Functions

How well do you understand functions? (On a scale from 1 – 5, 1 = not at all; 3 = somewhat; 5 = very well)		
Group	Δ Mean (Std Dev)	Mean (Std Dev)
1 (Arduino)	Increase 1.00 (1.07)	pre = 3.14 (0.64) post = 4.14 (0.83)
2 (Scratch)	Increase 0.45 (0.66)	pre = 2.18 (1.34) post = 2.64 (1.43)
3 (Arduino)	Increase 0.62 (1.86)	pre = 3.08 (1.14) post = 4.00 (0.91)
4 (Scratch)	Increase 0.50 (1.12)	pre = 2.57 (0.90) post = 2.67 (1.49)
5 (Both)	Increase 0.63 (1.11)	pre = 2.00 (1.17) post = 2.63 (1.13)
All	Increase 0.65 (1.13)	pre = 2.49 (1.22) post = 3.13 (1.35)

Table 4-15: Question 13 results summary – understanding functions

4.4.5. Question 14: Understanding Programming

How well do you understand computer programming? (On a scale from 1 – 5, 1 = not at all; 3 = somewhat; 5 = very well)		
Group	Δ Mean (Std Dev)	Mean (Std Dev)
1 (Arduino)	Increase 1.00 (0.53)	pre = 3.29 (0.70) post = 4.29 (0.45)
2 (Scratch)	Increase 0.44 (0.68)	pre = 1.91 (1.31) post = 2.36 (1.30)
3 (Arduino)	Decrease -0.31 (1.07)	pre = 3.38 (0.74) post = 3.33 (0.75)
4 (Scratch)	Increase 0.00 (0.58)	pre = 1.86 (0.83) post = 2.00 (1.00)
5 (Both)	Increase 0.94 (0.75)	pre = 1.42 (0.67) post = 2.21 (0.83)
All	Increase 0.47 (0.81)	pre = 2.25 (1.20) post = 2.73 (1.18)

Table 4-16: Question 14 results summary – understanding computer programming

4.5. Lab Completion

The post-survey asked participants to record which labs they completed and to what extent. On average, all groups came close to finishing each lab. Completion was on a scale from one to five; one meant the lab was not completed at all, four that the lab was completed, and five meant that the extra practice session was completed as well.

Which labs did you complete? (On a scale from 1 – 5, 1 = didn't do; 4 = finished; 5 = finished extra practice section, N/A = no attempt – if you were absent)						
	Mean (Std Dev)					
Group	Lab 1 – Blinking	Lab 2 – Dimmer	Lab 3a – Traffic Lights	Lab 3b – Push button	Lab 4 – Buzzer	Lab 5 – Temp Sensor
1 (Arduino)	4.86 (0.35)	4.14 (0.64)	3.86 (1.36)	N/A	3.29 (1.58)	4.00 (0.53)
3 (Arduino)	4.50 (0.65)	4.08 (0.76)	N/A	3.58 (0.86)	3.83 (0.80)	3.33 (1.37)
5 (Both)	3.67 (1.26)	3.67 (1.2)	N/A	N/A	4.05 (0.99)	N/A

Table 4-17: Summary of completion for Arduino labs

Which labs did you complete? (On a scale from 1 – 5, 1 = didn't do; 4 = finished; 5 = finished extra practice section, N/A = no attempt – if you were absent)					
	Mean (Std Dev)				
Group	Lab 1 – Moving Sprites	Lab 2 – MadLibs	Lab 3 – Tag Game	Lab 4 – Sound	Lab 5 – Calculator
2 (Scratch)	3.60 (1.02)	3.40 (1.11)	3.50 (1.12)	3.33 (1.15)	3.30 (1.27)
4 (Scratch)	3.20 (0.40)	3.20 (0.40)	3.20 (0.75)	3.60 (0.49)	4.00 (0.00)
5 (Both)	N/A	3.89 (1.33)	3.65 (1.34)	4.21 (0.97)	N/A

Table 4-18: Summary of completion for Scratch labs

4.6. Study Experience

4.6.1. Question 15 - 18: Easiness and Interestingness of Platforms

Questions 15 – 18 asked participants to rate how easy and how interesting they found Scratch or Arduino on a scale of one through five.

How interesting did you find learning Scratch / Arduino? (On a scale from 1 – 5, 1 = totally boring; 3 = ok; 5 = very interesting)				
How easy did you find learning Scratch / Arduino? (On a scale from 1 – 5, 1 = very difficult; 5 = very easy)				
	Easiness Mean (Std Dev)		Interestingness Mean (Std Dev)	
Group	Scratch	Arduino	Scratch	Arduino
1 (Arduino)	N/A	3.57 (0.73)	N/A	4.14
2 (Scratch)	2.91 (1.24)	N/A	2.55 (1.37)	N/A
3 (Arduino)	N/A	3.25 (1.01)	N/A	3.67 (0.75)
4 (Scratch)	3.25 (0.97)	N/A	3.13 (1.27)	N/A
5 (Both)	3.95 (1.00)	2.53 (1.19)	3.63 (1.22)	2.89 (1.29)
All	3.51 (1.17)	3.50 (1.09)	2.66 (1.28)	3.38 (1.16)

Table 4-19: Question 15 - 18 results summary – easiness and interestingness of platforms

4.6.2. Questions 19 - 20: Image of computing

Questions 19, 20a and 20b asked participants to report how learning either platform changed their image of computing. For Group 1, we asked only whether or not their

experience with Arduino changed their image of computing on a scale of one to five. (Question 20a). However, for Groups 2 – 5, we asked whether the experience changed their view of computing in a positive way, negative way, or not at all.

Did learning Scratch / Arduino change your image of computing?		
Yes – Positively Yes – Negatively No		
Group	Scratch	Arduino
1 (Arduino)	N/A	4.0
2 (Scratch)	No Change	N/A
3 (Arduino)	N/A	Positively / No
4 (Scratch)	No Change	N/A
5 (Both)	Positively	Positively
All	Positively / No Change	Unknown

Table 4-20: Question 19 - 20 summary – change in image of computing

4.6.3. Question 31: Likes

Questions 31 and 32 asked students whether there was anything in particular they liked or did not like in regards to the content of the sessions. Students answered the questions in prose, and their answers were then categorized into the categories below. The percentages shown represent the number of students reporting likes or dislikes in proportion to the other participants in their group who completed both the pre- and the post-survey.

Groups 1 and 3 had high percentages (over 85% and 45% respectively) of participants citing something about Arduino they enjoyed. The group that worked with both Scratch and Arduino had more participants reporting a preference for Scratch over Arduino.

Was there anything particular that you liked about the lectures / Scratch / Arduino?	
Group 1 (Arduino)	
Arduino	85.71%
Fun	14.29%
Interesting	14.29%
Group 2 (Scratch)	
Scratch	25.00%
Interesting	8.33%
Group 3 (Arduino)	
Arduino	46.15%
Fun	15.38%
Interesting	7.69%
Easy	7.69%
Familiar	7.69%
Learned	7.69%
Group 4 (Scratch)	
Scratch	50.00%
Fun	25.00%
Easy	12.50%
Group 5 (Scratch and Arduino)	
Scratch	36.84%
Fun	26.32%
Interesting	15.79%
Arduino	5.26%

Table 4-21: Question 31 – likes

4.6.4. Question 32: Dislikes

As with the "likes" section, we asked students what they did not like about the content of the course and categorized the students' prose. The three primary responses were "confusing," "boring," and "difficult." A number of students did not like particular aspects of Arduino, such as the programming environment or problems with getting their machines to recognize the Arduino device drivers. None of the Scratch groups reported disliking anything particular about Scratch as a programming environment.

Groups 2, 3 and 4 had particular challenges during the last session with the classroom situation. All five groups stated that they disliked how confusing the sessions were.

What did you NOT like about lectures / Scratch / Arduino?	
Group 1 (Arduino)	
Confusing	14.29%
Too fast	14.29%
Group 2 (Scratch)	
Confusing	25.00%
Boring	16.67%
Group 3 (Arduino)	
Confusing	23.08%
Arduino	23.08%
Difficult	23.08%
Group 4 (Scratch)	
Difficult	12.50%
Confusing	12.50%
Group 5 (Scratch and Arduino)	
Confusing	31.58%
Arduino	10.53%
Difficult	10.53%
Too fast	5.26%
Too particular	5.26%
Difficult	15.79%

Table 4-22: Question 32 – dislikes

A complete list of responses to Questions 31 and 32 can be found in Appendix E.

5. Conclusion

In this study, we produced two complete parallel courses in introductory programming concepts – one using Scratch for hands-on labs and one using Arduino. We piloted this course with five groups of high school students (119 students total) at two local schools during their regular class periods. The course materials are all freely available online [62] as a set of lecture slides and a set of five labs for each platform.

Overall, we cannot quantitatively answer our research question as to whether or not Arduino is an effective platform for teaching programming to novices based on our study. The results from the objective learning assessment questions (Questions 1 – 5) were mixed. On Question 1b, regarding use of basic variables, three of four groups had an increase in their scores, while one had no change. However, most students were able to answer this question correctly in the pre-survey. For Question 3 (conditionals), four out of five groups actually saw a decrease in their scores. Only one of the Scratch groups (Group 2) had an increase in their scores. And as for the concept of functions, all groups except for Group 3 had an increase in their scores on Question 5. Many students in Groups 1, 2, 4 and 5 who received a score of zero on the pre-survey, answered the question correctly on the post-survey. Since Group 3 mostly answered the question correctly in the pre-survey, their results on the post-survey had no significant change. In regards to Question 4 (iteration), no students in Groups 4 and 5 answered the question correctly in either pre- or post-survey. There was a significant increase in the number of students answering the question correctly in Group 1 (Arduino). Groups 2 and 3 had slight decreases in their mean score.

As far as the students' own assessment of their learning, most outside of Group 3 reported an increased understanding in all of the core programming concepts as well as programming as a whole. Since the majority of students in those groups had no previous programming experience, we raise two questions. First, how was it possible that with no experience, students still reported having some understanding of programming concepts? And following from that question, is it possible that in the pre-survey students' reporting understanding similarly named and related concepts from other fields, such as variables in mathematics? Possibly, the students did not have enough experience to correctly judge their own knowledge.

Overall, the Morro Bay High School students had a positive experience with Scratch and Arduino, though the only group at Morro Bay that did labs in Scratch was the group that did both Scratch and Arduino. There were a few students in Group 5 who were clearly frustrated in working with Arduino. This could be because of the driver difficulties they encountered – many spent 20 – 30 minutes of the last two sessions getting their computers to communicate with their Arduino boards.

The groups with previous programming experience (Groups 1 and 3) reported completing more of the labs. We observed the students in Group 3 expand upon the labs, even past the extra practice sessions. Groups 3 and 5 were also the groups that reported having a positive change in their image of computing from the sessions. Groups 2 and 4 reported no overall change. The most common complaints regarding the sessions regardless of lab platform was "confusing." Students did not always specify whether this was the lectures or the labs. The common likes regarding the sessions were "fun" and "interesting."

Based on in-class observations and discussions with the experienced high school teachers, we conclude that Arduino appears to be overly complicated for teaching introductory programming concepts to beginners. Both high school teachers, though, will continue to integrate more computing in to their classes. They intend to use Scratch in their computer applications classes and both Scratch and Arduino in their programming classes. The upper-level programming classes at San Luis Obispo high school were able to quickly pick up the Arduino IDE and language. Many of the students in the class are also on the school's robotics team and have discussed using the Arduino boards in their robotics projects. The teacher at Morro Bay High School plans on using the Arduino boards in the manufacturing concepts class after an introduction to computing through Scratch. He has also started a weekly "Maker Club" based on the Maker' Faire [63] that will meet to work on projects, including with Arduino boards. Both schools have purchased a classroom set of Arduino boards.

6. Future Work

We see multiple possible future additions to our work. First, it would be beneficial to reconcile the scope of the material presented with the time allotted in the classroom and rerun the study. This could be done by either expanding the time spent in the classroom or reducing the scope of the concepts covered. It may be appropriate to cover only a single concept in a study. Reducing the subjects covered may lessen the students' confusion with the material covered.

The study could also be replicated with stricter controls. One major variable was the experience level of each class. The high school teachers felt Arduino would be too complicated for their non-programming classes and requested that those classes be introduced to programming through Scratch first.

Instead of having each class use a single platform for the labs, each class could be split up so that half of the class worked with Scratch and half with Arduino. This was our original intention and the reason for having a common set of lectures. However, splitting the classes proved too complex for the given classroom situations.

With the inconclusive results from our survey, we believe an objective, quantitative assessment instrument needs to be created and validated to properly evaluate the course learning outcomes. While our total participant size was 119 participants, only 59 students completed both the pre-survey and post-survey (approximately 50%), and repeating a study with a validated instrument and more participants would be appropriate.

Bibliography

- [1] “Table of Contents, OOO Online, Spring 2012,” *Occupational Outlook Spring 2012*. [Online]. Available: <http://www.bls.gov/opub/ooo/2012/spring/home.htm>. [Accessed: 09-Apr-2012].
- [2] “Undergraduate CS Degree Production Rises; Doctoral Production Steady,” *Computing Research News - Online Archive - Computing Research Association*. [Online]. Available: http://www.cra.org/resources/crn-archive-view-detail/undergraduate_cs_degree_production_rises_doctoral_production_steady/. [Accessed: 28-May-2012].
- [3] “CS Principles,” *CS Principles*. [Online]. Available: <http://www.csprinciples.org/>. [Accessed: 28-May-2012].
- [4] “Table of Contents, OOO Online, Spring 2000,” *Occupational Outlook Spring 2000*. [Online]. Available: <http://www.bls.gov/ooo/2000/spring/contents.htm>. [Accessed: 31-May-2012].
- [5] Graves, Jada A., “The Best Jobs of 2012.” US News & World Report, 27-Feb-2012.
- [6] “Occupation Outlook Quarterly, Spring 2012.” [Online]. Available: <http://www.bls.gov/opub/ooo/home.htm>. [Accessed: 09-Apr-2012].
- [7] “Occupational Outlook Handbook, 2012-13 Edition -- Software Developers.” Bureau of Labor Statistics, U.S. Department of Labor.
- [8] “Employment Projections: 2010-2020 Summary,” *United States Department of Labor*. [Online]. Available: <http://www.bls.gov/news.release/ecopro.nr0.htm>. [Accessed: 05-Jun-2012].
- [9] “Computing Education & Future Jobs,” *NCWIT : Resources : NCWIT Resources : Computing Education & Future Jobs*. [Online]. Available: <http://www.ncwit.org/resources.res.computinged.html>. [Accessed: 27-Apr-2012].
- [10] “Accredited Program Search.” [Online]. Available: <http://main.abet.org/aps/Accreditedprogramsearch.aspx>. [Accessed: 05-Jun-2012].
- [11] CSTA Curriculum Improvement Task Force, “The New Educational Imperative: Improving High School Computer Science Education,” 2005.
- [12] “Computer Science Teachers Association - High School Surveys.”
- [13] “Running on Empty.” [Online]. Available: <http://csta.acm.org/runningonempty/>. [Accessed: 05-Jun-2012].
- [14] “Scratch Homepage.” [Online]. Available: <http://scratch.mit.edu/>. [Accessed: 09-Apr-2012].
- [15] “Overview \ Processing.org.” [Online]. Available: <http://processing.org/about/>. [Accessed: 09-Apr-2012].
- [16] J. Sarik and I. Kymissis, “Lab kits using the Arduino prototyping platform,” in *Frontiers in Education Conference (FIE), 2010 IEEE*, 2010, p. T3C-1 –T3C-5.
- [17] “Arduino Home Page.” [Online]. Available: <http://arduino.cc/>. [Accessed: 09-Apr-2012].
- [18] P. Jamieson, “Arduino for Teaching Embedded Systems. Are Computer Scientists and Engineering Educators Missing the Boat?”
- [19] A. Robins, J. Rountree, and N. Rountree, “Learning and teaching programming: A review and discussion,” *Computer Science Education*, vol. 13, no. 2, pp. 137–172, 2003.

- [20] L. McIver and D. Conway, “Seven deadly sins of introductory programming language design,” in *Software Engineering: Education and Practice, 1996. Proceedings. International Conference, 1996*, pp. 309–316.
- [21] R. D. Pea, “Language-independent conceptual ‘bugs’ in novice programming,” *Journal of Educational Computing Research*, vol. 2, no. 1, pp. 25–36, 1986.
- [22] “SIGCSE.” [Online]. Available: <http://www.sigcse.org/>. [Accessed: 20-Apr-2012].
- [23] “icer-conference.” [Online]. Available: <http://wp.acm.org/icer-conference/>. [Accessed: 20-Apr-2012].
- [24] “ITiCSE 2012, Haifa, Israel.” [Online]. Available: <http://www.iticse12.org.il/htmls/home.aspx>. [Accessed: 20-Apr-2012].
- [25] “ACE2012.” [Online]. Available: <http://www.sci.usq.edu.au/conferences/ace2012/>. [Accessed: 21-Apr-2012].
- [26] “Koli Calling - General Information.” [Online]. Available: <http://cs.joensuu.fi/kolistelut/>. [Accessed: 20-Apr-2012].
- [27] “Citrenz Conference 2012.” [Online]. Available: <http://www.naccq.ac.nz/>. [Accessed: 21-Apr-2012].
- [28] J. Sheard, S. Simon, M. Hamilton, and J. Lönnberg, “Analysis of research into the teaching and learning of programming,” in *Proceedings of the fifth international workshop on Computing education research workshop*, New York, NY, USA, 2009, pp. 93–104.
- [29] B. Adelson, “Problem solving and the development of abstract categories in programming languages,” *Memory & cognition*, vol. 9, no. 4, pp. 422–433, 1981.
- [30] M. de Raadt, R. Watson, and M. Toleman, “Teaching and assessing programming strategies explicitly,” in *Proceedings of the Eleventh Australasian Conference on Computing Education - Volume 95*, Darlinghurst, Australia, Australia, 2009, pp. 45–54.
- [31] A. Pears, S. Seidman, L. Malmi, L. Mannila, E. Adams, J. Bennedsen, M. Devlin, and J. Paterson, “A survey of literature on the teaching of introductory programming,” in *Working group reports on ITiCSE on Innovation and technology in computer science education*, New York, NY, USA, 2007, pp. 204–223.
- [32] L. E. Winslow, “Programming pedagogy--a psychological overview,” *SIGCSE Bull.*, vol. 28, no. 3, pp. 17–22, Sep. 1996.
- [33] P. Byckling and J. Sajaniemi, “A role-based analysis model for the evaluation of novices’ programming knowledge development,” in *Proceedings of the second international workshop on Computing education research*, 2006, pp. 85–96.
- [34] Simon, “Assignment and sequence: why some students can’t recognise a simple swap,” in *Proceedings of the 11th Koli Calling International Conference on Computing Education Research*, New York, NY, USA, 2011, pp. 10–15.
- [35] “Curricula Recommendations — Association for Computing Machinery.” [Online]. Available: <http://www.acm.org/education/curricula-recommendations>. [Accessed: 23-Apr-2012].
- [36] “Running on Empty: The Failure to Teach K-12 Computer Science in the Digital Age,” *CSTA : Running on Empty*. [Online]. Available: <http://csta.acm.org/runningonempty/>. [Accessed: 29-Apr-2012].

- [37] The CSTA Standards Task Force, “K-12 Computer Science Standards Revised 2011,” *K-12 Computer Science Standards Revised 2011*, 2011. [Online]. Available: http://csta.acm.org/Curriculum/sub/CurrFiles/CSTA_K-12_CSS.pdf. [Accessed: 28-Apr-2012].
- [38] O. Astrachan, R. Morelli, D. Barnette, J. Gray, C. Uche, B. Cowles, and R. Dovi, “CS principles: piloting a national course,” in *Proceedings of the 43rd ACM technical symposium on Computer Science Education*, New York, NY, USA, 2012, pp. 319–320.
- [39] O. Astrachan, T. Barnes, D. D. Garcia, J. Paul, B. Simon, and L. Snyder, “CS principles: piloting a new course at national scale,” in *Proceedings of the 42nd ACM technical symposium on Computer science education*, 2011, pp. 397–398.
- [40] D. D. Garcia, “The beauty and joy of computing (BJC), AP CS principles, and the CS 10K effort,” *J. Comput. Sci. Coll.*, vol. 27, no. 5, pp. 126–127, May 2012.
- [41] C. Kelleher and R. Pausch, “Lowering the barriers to programming: A taxonomy of programming environments and languages for novice programmers,” *ACM Comput. Surv.*, vol. 37, no. 2, pp. 83–137, Jun. 2005.
- [42] J. Maloney, M. Resnick, N. Rusk, B. Silverman, and E. Eastmond, “The Scratch Programming Language and Environment,” *Trans. Comput. Educ.*, vol. 10, no. 4, pp. 16:1–16:15, Nov. 2010.
- [43] Y. B. Kafai and M. Resnick, *Constructionism in practice: Designing, thinking, and learning in a digital world*. Lawrence Erlbaum, 1996.
- [44] M. Resnick, J. Maloney, A. Monroy-Hernández, N. Rusk, E. Eastmond, K. Brennan, A. Millner, E. Rosenbaum, J. Silver, B. Silverman, and Y. Kafai, “Scratch: programming for all,” *Commun. ACM*, vol. 52, no. 11, pp. 60–67, Nov. 2009.
- [45] C. M. Lewis, “How programming environment shapes perception, learning and goals: logo vs. scratch,” in *Proceedings of the 41st ACM technical symposium on Computer science education*, New York, NY, USA, 2010, pp. 346–350.
- [46] O. Meerbaum-Salant, M. Armoni, and M. Ben-Ari, “Habits of programming in scratch,” in *Proceedings of the 16th annual joint conference on Innovation and technology in computer science education*, 2011, pp. 168–172.
- [47] M. Rizvi, T. Humphries, D. Major, M. Jones, and H. Lauzun, “A CS0 course using Scratch,” *J. Comput. Sci. Coll.*, vol. 26, no. 3, pp. 19–27, Jan. 2011.
- [48] J. H. Maloney, K. Peppler, Y. Kafai, M. Resnick, and N. Rusk, “Programming by choice: urban youth learning programming with scratch,” in *Proceedings of the 39th SIGCSE technical symposium on Computer science education*, New York, NY, USA, 2008, pp. 367–371.
- [49] O. Meerbaum-Salant, M. Armoni, and M. M. Ben-Ari, “Learning computer science concepts with scratch,” in *Proceedings of the Sixth international workshop on Computing education research*, 2010, pp. 69–76.
- [50] P. Bender and K. Kussmann, “Arduino based projects in the computer science capstone course,” *J. Comput. Sci. Coll.*, vol. 27, no. 5, pp. 152–157, May 2012.
- [51] P. Jamieson, “Arduino for Teaching Embedded Systems. Are Computer Scientists and Engineering Educators Missing the Boat?,” in *International Conference on Frontiers in Education: Computer Science and Computer Engineering*, 2011.

- [52] R. Balogh, "Educational robotic platform based on arduino," in *Proceedings of the 1st international conference on Robotics in Education, RiE2010. FEI STU, Slovakia*, 2010, pp. 119–122.
- [53] D. J. Malan and H. H. Leitner, "Scratch for budding computer scientists," in *Proceedings of the 38th SIGCSE technical symposium on Computer science education*, New York, NY, USA, 2007, pp. 223–227.
- [54] U. Wolz, H. H. Leitner, D. J. Malan, and J. Maloney, "Starting with scratch in CS 1," *ACM SIGCSE Bulletin*, vol. 41, no. 1, pp. 2–3, 2009.
- [55] U. Wolz, J. Maloney, and S. M. Pulimood, "'scratch' your way to introductory cs," *ACM SIGCSE Bulletin*, vol. 40, no. 1, pp. 298–299, 2008.
- [56] "Alice.org," *Alice.org*. [Online]. Available: <http://alice.org/>. [Accessed: 28-May-2012].
- [57] "San Luis Obispo High," *National Center for Education Statistics - Common Core of Data*. [Online]. Available: http://nces.ed.gov/ccd/schoolsearch/school_detail.asp?Search=1&DistrictID=0634800&ID=063480005880. [Accessed: 25-Apr-2012].
- [58] "California Association of Regional Occupational Centers and Programs," *CAROCP*. [Online]. Available: <http://www.carocp.org/>. [Accessed: 30-Apr-2012].
- [59] California Department of Education, "2011 Adequate Yearly Progress Chart -- San Luis Obispo High School," *2011 Adequate Yearly Progress Chart*. [Online]. Available: <http://api.cde.ca.gov/Acnt2011/2011APRSchAYPReport.aspx?cYear=2010-11%20&allcds=40-688094036703>. [Accessed: 30-Apr-2012].
- [60] "Morro Bay High," *National Center for Education Statistics - Common Core of Data*. [Online]. Available: http://nces.ed.gov/ccd/schoolsearch/school_detail.asp?Search=1&DistrictID=0634800&ID=063480005875. [Accessed: 25-Apr-2012].
- [61] California Department of Education, "2011 Adequate Yearly Progress Chart -- Morro Bay High School," *2011 Adequate Yearly Progress Chart*. [Online]. Available: <http://api.cde.ca.gov/Acnt2011/2011APRSchAYPReport.aspx?allcds=40688094034807>. [Accessed: 30-Apr-2012].
- [62] "Teaching Introductory Programming Concepts: A Comparison of Scratch and Arduino." [Online]. Available: http://www.anniebeug.com/thesis/My_Thesis/About.html. [Accessed: 05-Jun-2012].
- [63] "Maker Faire - Maker Faire Bay Area 2012 May 19th & 20th." [Online]. Available: <http://makerfaire.com/>. [Accessed: 05-Jun-2012].

Appendix A: Lectures

Session 1

What is this?

- Master's thesis in computing education
- 5 sessions
 - short lecture
 - lab
- Comparing Scratch with Arduino

What is a computer?

- Data processor
- Problem solving tool
- Task automator
- Where can you find computers?

What is a program?

- A sequence of steps that tell a computer what to do
- Programs express "algorithms"

What is an algorithm?

- An algorithm is a set of steps taken to solve a problem or task
- An algorithm is a sequence of instructions for performing some calculation

What is an algorithm?

- Wash your hair
 - Get in the shower
 - Turn on the water
 - Get your hair wet
 - Open the shampoo bottle
 - Pour some in to your hand
 - Rub it in to your hair
 - Rinse your hair
 - Repeat if desired

What is an algorithm?

- Find the average of a list of numbers
 - Sum up all the numbers
 - Count the total number of numbers
 - Divide the sum of the numbers by the count of the numbers

What is software vs. hardware?

- Hardware is the physical components
 - What are some different types of hardware?
- Software is the programs that run on top of the hardware
 - What are some different types of software?
- The lines between software and hardware can get blurry

What is source code?

- Source code is just the files that programmers write that makes up the program
- You'll be creating source code in this class!

Session 2

What is a variable?

- Name for a storage location
- Think of a storage bin
- Data (some piece of information) can be put in the variable, and retrieved later

What is assignment?

- Assignment sets the value of variable
- The value (say a number or a word) is put in to the variable
- The variable is a storage location for the value

What is assignment?

FavoriteNumber = 13

What is a variable?

- You can use the variable more than once

```
a = 7 + 3
(a is now 10)
a = 1 + 3
(a is now 4)
b = 5
a = b + 1
(a is now 6)
```

What is code?

- Code does things
 - Assignment
 - `a = 2`
 - Evaluate expressions
 - `10 * 3 + 1`

What is code?

```
Number x = 10
Number y = 2
Number z = x + y
Print "Hello"
(Prints out Hello)
Print z
(Prints out 10)
```

- Declarations of variables, `x`, `y`, and `z`
- Assignment of 10 to `x`, 2 to `y`, and `x + y` to `z`
- `x + y` is an expression that produces the number 12
- Print "Hello" is a statement which will print out the word "Hello"

What can variables represent?

- In algebra, variables represent numbers
- In programming, variables can represent

number	1, 5, 4.2, 39402394
letter	a, b, c, d, e
strings of letters	"Hello, my name is Annie."
dates / time	Saturday, January 14 th , 2012 3:04 AM
booleans	True or False

What is assignment?

`MyName = "Annie"`

What are constants?

- Constants are variables that once set, aren't changed

What are collections?

- Lists
 - Lists are collections of variables in a given order
 - Lists can also be called "arrays"

Index	Value
0	hotdogs
1	buns
2	ketchup
3	mustard
4	relish
5	soda

What are collections?

The diagram illustrates a collection of food items. A table with two columns, 'Index' and 'Value', contains the following data:

Index	Value
0	hotdogs
1	buns
2	ketchup
3	mustard
4	relish
5	soda

An orange arrow labeled 'Indexes' points to the 'Index' column, and another orange arrow labeled 'Values' points to the 'Value' column.

Session 3

What is a conditional?

- A prerequisite
- Something that must be true for something else to happen

Remember booleans?

- Either "true" or "false"
- "Yes" or "no"
- Is the sky blue?
 - True or false? True!
- Is $3 > 5$?
 - True or false? False!

What is a conditional?

- In programming, we can say

```
If something is true
Then
    do one thing
Else (something is false)
    do another thing
```

What is a conditional?

- We can have compound conditions
 - AND → both conditions have to be true
 - OR → one condition has to be true

```
3 > 4 AND 3 > 2 → false
3 > 4 OR 3 > 2 → true
3 < 4 AND 3 > 2 → true
```

What is a conditional?

```
a = 4
If a > 5
    print "A is greater than 5"
Else if a >= 4
    print "A is 4 or 5"
Else
    print "A is less than 4"
```

What does this print out?

What is a conditional?

```
a = 4
if (a > 5 AND a < 10)
    print "A is 6 to 9"
else
    print "A is 5 or less or 10
    or greater"
```

What does this print out?

What is a conditional?

```
a = 4, b = 10
if (a >= 5 OR b > 10)
    print "A is greater than 5 OR
    b is greater than 10"
else
    print "A is less than 5 AND
    B is less than 11"
```

What does this print out?

Session 4

What is iteration?

- Repeating
- Doing something multiple times
- "a procedure in which repetition of a sequence of operations yields results successively closer to a desired result" (Merriam-Webster dictionary definition)
- "loop"

What is iteration?

- Think of
 - Running around a track
 - Knitting a scarf
 - Finding the factors of a number
 - Drum loops

Remember conditionals?

```
a <= 10
b == c
x + y < 100 OR x - z < 1000
```

What is iteration?

- We can use conditionals to control the repetition of loops

```
Loop while x < 5
    print "Hello"
    x = x + 1
End Loop (when x is set to 5)
```

Output is
Hello Hello Hello Hello

What is iteration?

```
sum = 0
currNum = 1
Loop while currNum < 10
    sum = sum + currNum
    currNum = currNum + 1
End Loop
```

What is iteration?

```
index = 0
array = hotdogs, chocolate
       cakes, cornbread, mashed
       potatoes
Loop while index < number of
    items
    print "I love" array->index
    index = index + 1;
End Loop
```

Session 5

What is a function?

- Smaller units of code
- Re-use
- Easier to read
- Named
- Hide details

What is a function?

- Pass in variables
- Return variables

What is a function?

- What does a function look like?

```
int average(int a, int b, int c)
{
    int sum = a + b + c;
    int ave = sum / 3;
    return ave;
}
```

What is a function?

- Think of a soda machine!

- Input?
- Output?

What is a function?

- What does this "function machine" do?

Input	Output
1	5
2	10
3	15
4	20

Scratch Lecture

Scratch

Scratch

Arduino Lecture

Introduction to Arduino

- Arduinos are small computers
- You can write your own software to run on them
- You can hook them up to other parts
- The Arduino is the hardware...we'll provide the software

Introduction to Arduino

Introduction to Arduino

Program Structure

- Comments at the top
- Constant variables
- Setup
- Loop

Comments

- Comments at the top
 - Purpose
 - Author, date
- Constant variables
- Setup
- Loop

```
/**
 * This is my great code!
 * By: Annie
 * On: 5/4/2011
 **/
```

Constants

- Comments at the top
- Constant variables
 - Values which will stay the same
- Setup
- Loop

```
int FIRST_PIN = 1;
int SECOND_PIN = 2;
int THIRD_PIN = 3;
int DELAY = 2000;
```

Setup

- Comments at the top
- Constant variables
- Setup
 - Set up the pins
- Loop

```
void setup() {  
  // Initialize the pins  
  pinMode(FIRST_PIN, OUTPUT);  
}
```

Loop

- Comments at the top
- Constant variables
- Setup
- Loop
 - Perform the interaction over and over

```
void loop() {  
  digitalWrite(FIRST_PIN, HIGH); // turn LED on  
  delay(DELAY); // wait for a second  
  digitalWrite(FIRST_PIN, LOW); // turn LED off  
  delay(DELAY); // wait for a second  
}
```

Conditionals: = vs. ==

=	Assignment	x = 5 (sets x to 5)
==	Comparison	X == 5 (compares x to 5 → true!)

Conditionals

```
int myNum = 0;  
  
if (myNum == 0) { // == means comparison  
  myNum = 1; // = means assignment  
} else if (b == c) {  
  
} else {  
  
}
```

More Comparisons

==	Equals
!=	Not equals
>	Greater than
<	Less than
>=	Greater than or equal
<=	Less than or equal

Loops

- You're already using one!
- Remember

```
void loop() {  
  digitalWrite(FIRST_PIN, HIGH); // turn LED on  
  delay(DELAY); // wait for a second  
  digitalWrite(FIRST_PIN, LOW); // turn LED off  
  delay(DELAY); // wait for a second  
}
```

Loops

- For loops are useful for going through arrays
- You can use them inside of the main "loop"

```
void loop() {  
  int array = { 3, 6, 2, 1 };  
  int arraySize = 4;  
  int sum = 0;  
  
  for (int i = 0; i < arraySize; i = i + 1) {  
    sum = sum + array[i];  
  }  
}
```

Functions

- loop, digitalWrite, and delay are all functions!

```
void loop() {  
  digitalWrite(FIRST_PIN, HIGH); // turn LED on  
  delay(DELAY);                  // wait for a second  
  digitalWrite(FIRST_PIN, LOW);  // turn LED off  
  delay(DELAY);                  // wait for a second  
}
```

Functions

```
void loop() {  
  int num = 12;  
  int result = myFunction(num);  
}  
  
int myFunction(int anInt) {  
  int newNum = anInt + 4;  
  return newNum;  
}
```

Functions

- What is "void"?

```
void loop() {  
  int num = 12;  
  int result = myFunction(num);  
}
```

Appendix B: Scratch Labs

Session 1

Introduction

The goal of this first lab is to introduce you to the Scratch programming environment (Scratch IDE). You will learn how to create a basic Scratch program. Please review the glossary at the end of this write-up to learn the terms specific to Scratch programming. There is a glossary at the end of this write-up to review.

Step 1: Open the Scratch IDE

Open the Scratch IDE. The icon is shown in Figure 1. You can find it either in the Start Menu or on the Desktop.

Figure 1: Scratch IDE icon

Step 2: Open the file

Open the Scratch file "IntroToProgramming/Labs/Scratch/Lab1/CatSpin.sb". You should see a stage like in Figure 2. Sprites are the images you place on the stage. You will see in this Scratch program one sprite so far—the same cat from the Scratch icon.

Figure2: Stage area

Figure 3 shows the script area of Scratch. This is where you will write your program to control the sprites and the stage.

Figure 3: Script for cat sprite

Step 3: Run the program

To run the script, click on the green flag (as seen in the top-right corner of Figure 2). To stop the script, click on the red circle, which is next to the green flag. As you can see in the script in Figure 3, the script starts when the green flag is clicked.

Step 4: Learn about Scratch

You can add new blocks by clicking on them in the block area (Figure 4 – you will see different blocks depending on which block category you have selected) and dragging them in to the script area. The blocks click together (like Legos) to run as a whole script. You can remove blocks from the script by clicking on them and dragging to the block area or by right-clicking on them and selecting "delete".

Figure 4: Block area

The blocks are organized into different categories (as seen in Figure 5). Click on different categories to see the blocks within.

Figure 5: Block categories

Look through each different category. You will be putting together different blocks from different categories to make your script.

Category Name	Description
Motion	Move <i>sprites</i> around the <i>stage</i> .
Looks	Change the appearance of the <i>sprite</i> .
Sound	Play sounds—control what sounds, tone, volume, tempo.
Pen	Draw on the <i>stage</i> similar to a paint program.
Control	Control the flow of the <i>script</i> —make decisions about what to do and how many times to do it.
Sensing	Interact with the user and other <i>sprites</i> .
Operators	Perform math and comparisons.
Variables	Create placeholders for values (such as text and numbers).

Figure 6: Different parts of the Scratch window

Step 5: Change the Code

- First, add a new sprite to the stage. In the sprite area of Scratch, click the open sprite button and choose a new sprite.
- In the sprite area, click on the sprite for which you want to edit the script. The script for that sprite will appear in the script area.
- Start adding blocks to the script for your new sprite. Try making it move around, rotate, even change colors (look in the Looks category).

Step 6: Extra Practice

Change the stage

In the sprite area, click on the stage. Back in the script area, click to the "backgrounds" tab. Under "new background" you can choose to either import an existing background or paint your own. Switch the stage to something other than a blank white screen. Notice that when you clicked the stage in the sprite area, the blocks in the script area changed. You can have a script just for the stage. For instance, you could have the stage change background. Find the block to do this and add it to the stage's script.

Change a sprite's costume

In the same way you can change the stage's background, you can also change "costumes" for the sprites. Select the cat sprite. At the top of the script area, click on the "costumes" tab. Add a new costume for the cat. Then have the script for the cat switch between the costumes. You can use costumes to make the cat appear to be walking.

Glossary

<i>block</i>	A block is a piece of behavior as part of a <i>script</i> . Blocks can control one of the <i>sprites</i> on the <i>stage</i> or the <i>stage</i> itself.
<i>script</i>	A script is a program in Scratch. It is comprised of <i>blocks</i> . A script acts on a <i>sprite</i> .
<i>sprite</i>	A sprite is an actor. Sprites appear on the stage. Sprites can be controlled by <i>blocks</i> and <i>scripts</i> .
<i>stage</i>	The stage is where the sprites can be seen. The behavior of the <i>stage</i> can be controlled using <i>blocks</i> and <i>scripts</i> .

Session 2

Introduction

The purpose of this second lab is to learn how to use variables in Scratch. To do this, you will create an interactive MadLibs game.

Step 1: Open the Scratch Program

Open the Scratch file "IntroToProgramming/Labs/Scratch/Lab1/MadLibs.sb". You will create variables, all strings. Each variable will hold a different word for the MadLib. Examples of the word types are nouns, verbs, adjectives, adverbs, exclamations, place names, people names, articles of clothing, etc.

You will have one character ask the other character for a value for each word. Then you will substitute those words in to a story.

For example, if your words were:

Noun	tangerine	Adjective	twisted
City	Los Osos	Verb	scream

Your story could be:

Once upon a time in **Los Osos**, a young student named Wanda found a magic **tangerine**. She used her magic tangerine to go on a **twisted** adventure. On this adventure, she learned how to **scream**....

But maybe a little longer and more creative ;-)

In summary, what you will do is:

1. Create the variables
2. Ask the player to enter values for each variable
3. Create a story based on the variables
4. Display the story

The blocks you will want to use are:

- variables, set variable, show variable
- ask
- say
- broadcast, when I receive
- when "green flag" clicked, stop script
- join (to join two strings together)

Figure 1: MadLibs Program in Scratch

Session 3

Introduction

The purpose of the third lab is to learn how to use conditionals in Scratch to make decisions and sense what is going on in the program. To do this, you will create a game of tag between a cat and a crab.

Step 1: Open the Scratch Program

Open the Scratch file "IntroToProgramming/Labs/Scratch/Lab3/ CatCrabTag.sb" from the Desktop. Use the blocks in the Motion category to move the sprites around. Move the sprites based on what keys are being pressed. The block for knowing which key has been pressed is in the Sensing category. For instance, for the first sprite, you could use A to be move left, D to be move right, S to be move down and W to be move up. Your script should also sense whether or not the two sprites are touching (look at the Sensing category). If they are, have one of them say something, like "Bump!".

Figure 1: CatCrabTag file in Scratch

The blocks you will want to use include

- Control
 - when clicked
 - forever (loop)
 - if
- Looks
 - say
- Sensing
 - key pressed
 - touching
- Motion
 - change x by

- change y by

Step 2: Extra Practice

Add an extra key input to have the sprites spin.

Next, add a script to the stage that senses where the sprites are on the stage. If a sprite goes in to one of the corners, move it directly to the opposite corner.

Use your imagination! You can make this game as intricate as you want.

Session 4

Introduction

The purpose of this second lab is to learn how to use loops and lists in Scratch. To do this, you will create a song as a list of notes. Then, you will have the sprites in your project perform the songs.

Step 1: Open the Scratch Program

Open the Scratch file "IntroToProgramming/Labs/Scratch/Lab4/ MySong.sb".

Step 2: Change the Code

The blocks you will want to use include:

- Control
 - repeat
- Sound
 - play note
 - set instrument
- Variables
 - make a list
 - make a variable
 - set variable
 - item of list

You will need to create a list to hold all your notes. You will also want another variable that keeps track of which note in the list you are playing—this is called an index. Each time through the loop, you will need to increment (add 1 to) your index.

You can also display the values of your variables on the stage by clicking the check box next to your variable in the blocks area.

Figure 1: Template for song program

Step 3: Add More Features

Once you have your song playing through once, add more features to your script.

- Have the instrument sent to a random instrument.
- Ask the user how many times they want to play the tune.

Step 4: Extra Practice

Again, you can use your imagination and do anything you want!

- Have the script start when one of the sprites is clicked.
- Add motion to your project – animate the sprites in time to the song.
- Add a second tune as harmony to the first.

Session 5

Introduction

In this lab, you will use a new application—Build Your Own Blocks (BYOB)—to create a program. BYOB is an extension to Scratch, written at UC Berkeley. BYOB adds on new features to Scratch, namely functions. In BYOB, you can create your own blocks, which act like functions. Functions are chunks of code that you can re-use over and over again in different parts of your code. They are like templates for something you will want to do over and over again. Think of finding the average of a list of numbers—you'll use the same steps (sum all numbers then divide by the number of numbers) every time, but on different lists of numbers.

Step 1: Open the BYOB Program

In BYOB (instead of Scratch) open the file "IntroToProgramming/Labs/Scratch/Lab5/MyCalc.ypr". There is a new block created just for this lab, called "MySum". It will show you how you can create your own functions.

You will create your own calculator functions

- Average – of a list of numbers
- Max – find the largest number in the list

Step 2: Create Your Blocks

Here is a detailed outline of how the MySum block was created. Use this as a guide as how to create your two new blocks.

1. At the bottom of the Variables category, click on the "Make a block" button.
2. In the "Make a block" dialog box that pops up
 - a. Select a category ("Operators")
 - b. Select a block type ("reporter" – this block will return a value)
 - c. Give your block a name ("MySum")

3. In the block editor, build your new block like you would a regular script for a sprite
 - a. Add input to your block
 - i. Your input will be the list of numbers you will want to send to your function
 - ii. Click on the "+" plus sign next to your block name

- iii. Select "input name" and click the black triage next to it
- iv. For both functions, you will want your input to be a list of numbers

- v. Now, whenever you want to use the list ("MyList") passed in, you can just drag it from the top block in to your script

- b. Add variables to be used within your block by using the "script variables" block in the "Variables" category
 - i. You can create new script variables by clicking on the right triangle
 - ii. Double click on the new variable to rename it
- c. You can use both your input variables ("MyList") and local variables ("index") in your blocks

- d. Finally, send the result you want back out of the block by using the report block

Step 3: Use Your Block

1. Once you click "ok", your block will show up in the category you selected; in this example, "Operators"
2. Since you have a reporter block (which returns a value) you can use it in any block that takes a reporter block
3. In this example, the value of the block is set to a variable

Step 4: Extra Practice

Write a function (block) to sort the items in a list in ascending order. For example, if you are given the list

3, 4, 2, 19, 4, 22, 0, 1

your function should return a list

0, 1, 2, 3, 4, 4, 19, 22

Now rewrite the max function using sort; sorting the list will change the location of the maximal value.

Appendix C: Arduino Labs

Session 1: Introduction

Introduction

The goal of this first lab is to familiarize you with Arduino. Arduino is an opensource microcontroller. It can be used to run programs that you write. These programs can interact with the environment – make lights blink, read temperature sensors, spin motors.

Arduino refers to two things. First, to the programming environment we use to control the boards, called the Arduino IDE. And also to the Arduino board itself.

Step 1: Open the Arduino IDE

Open the program labeled "Arduino" from the desktop. The icon can be seen in Figure 1.

Figure 1: The Arduino icon

Step 2: Open the program

From the Lab 1 folder (on the desktop, "IntroToProgramming/Arduino/Labs/Lab1") open the Blink.ino file in the Blink folder. This is the source code we will be using for the lab.

Once you have opened it, it will look similar to the window in Figure 2.

Figure 2: Code in the Arduino IDE

Now you can verify the code by hitting the verify button. In the bottom section of the Arduino IDE, you should see a message like the one in Figure 2 that says **"Done Compiling. Binary sketch size...."**. If you see something else, try verifying a second time. If you still see an error, then you may have a mistake in your code. Try grabbing a fresh copy of the lab and make sure nothing gets changed.

Figure 3: The verify button

Step 3: Load the board

Now you can upload the code you just wrote so that it runs on the Arduino board.

With the USB cable connected from the computer to the Arduino board, hit the upload button. If you get an error that says **"avrdude: stk500_getsync(): not in sync: resp=0x00"** you probably need to select a different port for the Arduino. To do this, go to the menu Tools → Serial Port and select a different option. Then try uploading again. Make sure your Arduino is plugged in to the correct USB port. If you still get an error, you may need to reload part of the Arduino software. This is described in ["IntroToProgramming/Labs/Arduino/Setup/Arduino-setup.pdf"](#).

Figure 4: The upload button

Figure 5: Arduino board with transfer lights

Once you have pushed the upload button, you should see a couple lights on the Arduino board blink (area B in Figure 4). These mean your code is transferring to the board.

The board should look like Figure 6.

Figure 6: Arduino board with LED and resistor attached to breadboard

Step 4: Change the code

Change the code so that the light blinks much more slowly. To do this, you will update the code and then load it on to the board.

Step 5: Comment your code

Comments are part of your code that does not get run as part of the program. Instead, they serve to describe your code. Add comments like in Figure 7 to the top of your code describing what you wrote. Comments either start with two slashes ("//") or are between a slash-star ("/**" and "**/"). Figure 7 shows the second type of comment. These are usually used at the very beginning of code to describe your whole program and to say the author and creation date. Comments like Figure 8 are used inside your code to describe what the lines near the comment do. These can be either on their own line or at the end of another line.

```
/**
 * Lab 1: Introduction
 *
 * This program causes a light connected to the specified pin
 * to blink off and on at intervals defined by the BLINK_LEN
 * variable. BLINK_LEN specifies the number of milliseconds
 * (multiply by 1000 to get seconds).
 *
 * asbeug@calpoly.edu
 * Fall 2011 - Spring 2012
 */
```

Figure 7: Comments at the top of the program

```
//This is a comment on its own line
digitalWrite(BLINK_PIN, HIGH);    // Comment at end of a line
```

Figure 8: Comments in the code

Step 6: Extra Practice

Make another set of changes to the code so that the LED blinks out SOS. The sequence for SOS is short, short, short, long, long, long, short, short, short.

Glossary

<i>breadboard</i>	A breadboard is an electronics component that is used to hook wires up to other electronics components. Breadboards allow projects to be put together without soldering wires together.
<i>LED</i>	An LED is a small light. It allows electricity to flow through it one way. When it is hooked up to electricity properly, it glows.
<i>library</i>	A library is a bundled, re-usable piece of software.
<i>microcontroller</i>	A microcontroller is just a small computer on a single board. It does not have a keyboard, mouse, or monitor, but will have other input and output methods.
<i>opensource</i>	Opensource means that the source of a program (or the designs for a microcontroller) is shared.
<i>resistor</i>	A resistor restricts the flow of electricity.
<i>sketch</i>	A sketch is an Arduino program used to control an Arduino microcontroller board.

Session 2: Variables

Introduction

The purpose of this lab is to learn about variables. We will be creating a dimmer switch using an LED and what is called a potentiometer. A potentiometer lets you change how much electricity goes to the LED, which will change the brightness of the LED. Turning the potentiometer up (usually by turning the dial or sliding the slider to the right), increases the flow of electricity through it; turning the potentiometer down decreases the flow of electricity through it.

Step 1: Load the sketch

From the desktop, open the program called "IntroToProgramming/Arduino/Lab2/LEDDimmer/LEDDimmer.ino" in the Arduino IDE. This is the base sketch you will use to create your own program. Your board should be wired up like Figure 1. When you load the sketch on to your board, you should see the LED on.

Figure 1: Arduino board with LED and potentiometer

Step 2: Change the code

Your task in this lab is to make the potentiometer control the brightness of the LED.

Read through all the code!

Add code to the places where it says "TO DO" in the comments (`// TO DO`)

You will need to have one variable to keep track of the input from the potentiometer. You will need to have another variable to tell how bright to set the LED—which will depend on the state of the potentiometer.

In Lab 1, when we made the LED blink off and on, we used a *digital* write. This sent either HIGH or LOW to the LED. HIGH is equivalent to "on", and LOW is equivalent to "off". For this lab, we want to read and write an analog signal. The analog signal will allow us to use more than just two values (more than just off or on).

When we read from the potentiometer, the analog read function will return a value between 0 and 1024. When we write to the LED using the analog write function, we send it a value between 0 and 255, where 0 is completely off and 255 is full brightness.

Here are the functions you will need to use to read from the potentiometer and write to the LED. Instead of using the variables `some_analog_pin` and `brightness_number`, substitute

your own variables.

```
myResult = analogRead(some_analog_pin);  
analogWrite(some_analog_pin, brightness_number);
```

You can write the values of the variables out to the serial monitor. To do this, use the following function.

```
Serial.println(brightness_number);
```

With the program running on your board and your board connected to your computer, start the serial monitor by clicking on the icon at the top right of the Arduino IDE window or from the Tools → Serial Monitor menu.

Step 3: Extra Practice

There is a way to write this program with fewer variables. A variable is just a name for a storage location of a value. And functions return values, which can be used directly or assigned to variables. When we say

```
myResult = analogRead(some_analog_pin);
```

we are assigning the result of a call to the `analogRead` function to the variable called `myResult`.

Glossary

<i>analog</i>	A continuous signal across a whole range of values.
<i>digital</i>	A signal that is either on or off.
<i>potentiometer</i>	A potentiometer is a resistor. It has a control on it, so you can vary the resistance.

Session 3a: Conditionals – Traffic Light

Introduction

In this lab, we'll practice using conditionals (if-else statements) by simulating a traffic light using LEDs.

Step 1: Open the Sketch

Open the sketch "IntroToProgramming/Arduino/Lab3/TrafficLight/TrafficLight.ino" from the desktop.

Your board should look like Figure 1.

Figure 1: Board setup

Step 2: Change the Program

Fill in the part of the code that will determine the conditions for changing the traffic light – from green to yellow to red. Have the green and red lights stay on for 4 seconds and the yellow light stay on for 2 seconds.

Step 3: Extra Practice

Add a pushbutton and “walk” light LED to your breadboard and wire it up. Now, update the code so that the “walk” light turns on after the walk button has been pushed and the traffic light is red. The “walk” light should turn off when the light switches back to green.

For this step, your board should look like Figure 2.

Figure 2: Board setup with walk button

Session 3b: Conditionals – Pushbutton Light

Introduction

In this lab, we'll practice using conditionals (if-else statements) by simulating an on-off light switch. When you push the button, the light will turn on. When you let go, the light will turn off.

Step 1: Open the Sketch

Open the sketch "IntroToProgramming/Arduino/Lab3/LightSwitch/ LightSwitch.ino". Your board should look like Figure 1.

Figure 1: Board setup

Step 2: Change the Program

Fill in the part of the code that will determine the conditions for turning the light on and off. To do this, you will need to use conditionals (if statements) to test whether or not the button is being pressed.

To read the state of the button (whether it is off or on) you will use:

```
digitalRead(BUTTON_PIN);
```

The syntax for an if statement is

```
if ( x == HIGH ) {  
    y = HIGH;  
} else {  
    y = LOW;  
}
```

Notice that you use the double equals sign (==) for comparison and the single equals sign (=) for assignment. Instead of comparing the variable x to HIGH, you'll want to compare the output of the read function to HIGH.

And instead of assigning a value to the variable `y`, you'll want to turn on or off the LED. (See Lab 1 or 2 to see how to do this.)

Step 3: Extra Practice

Add a second LED. Have the one push button control both LEDs. Have one LED turn on and the other turn off when the push button is pushed.

Session 4: Iteration

Introduction

In this lab, you'll create a tune with a small speaker (called a Piezo buzzer). You'll create arrays of notes to represent the tune.

Step 1: Open the Sketch

Open the program labeled "IntroToProgramming/Arduino/Lab4/PiezoBuzzer/PiezoBuzzer.ino". Your board should look like Figure 1.

Figure 1: Board setup

Step 2: Change the Code

You will need to fill in all the parts of the code marked `"//TO DO"`. These include:

- Add variables
 - the pin the speaker is connected to
 - the number of milliseconds to play the note for
 - the number of milliseconds to wait before playing the next song
 - the number of milliseconds to wait at the end of playing all the notes before

starting over again

- Setup the pin for the buzzer for OUTPUT (see Lab 1 for reference on how to do this)
- Create a loop for playing all the notes. It should look similar to this, but with your own variables and the appropriate timing (delays):

```
for (int i = 0; i < numNotes; i++) {  
  // Get the note from the array  
  int myNote = array[i];  
  
  // Call function to play the note  
  buzz(myNote, NOTE_DURATION);  
}
```

Step 3: Extra Practice

Add a second song to play after the first song.

Add a push button to control the song playing – only play the sound after the button has been pushed. (Review Lab 3 for how to use the push button.)

Session 5: Functions

Introduction

In this lab, you'll write and use functions to create a digital thermometer.

Step 1: Open the Sketch

Open the program labeled "IntroToProgramming/Arduino/Lab5/TemperatureSensor/ TemperatureSensor.ino" from the desktop.

Figure 1: Board setup

Step 2: Verify the Libraries are Available

Libraries are programs that other people have written that you can use in your code.

Libraries contain variables and functions that you can call from your source code. For this lab, we'll use two libraries to talk to the temperature sensor. You'll have to make sure that your code can find these libraries. First, you need to make sure you have the following files:

```
My Documents\Arduino\libraries\DallasTemperature\DallasTemperature.cpp
My Documents\Arduino\libraries\DallasTemperature\DallasTemperature.h
My Documents\Arduino\libraries\OneWire\OneWire.cpp
My Documents\Arduino\libraries\OneWire\OneWire.h
```

If you don't have those files, you can copy them over from the Lab5 folder.

The "#include" lines tell the compiler to look at the libraries when it is compiling your code. These should already be in the lab code.

There is a second sketch "IntroToProgramming/Arduino/Lab5/TemperatureSensor/TemperatureSensor.ino" which you can use to test the functioning of your board and sensor.

Step 3: Change the Code

In the main loop function, a function is called to get the temperature. The result from that call (the temperature from the sensor in Celsius) is passed to a function that you will fill in.

The function you'll be completing is called `convertCelsiusToFahrenheit`—look for this at the bottom of the sketch. This function should calculate the equivalent degrees in Fahrenheit based on the value in Celsius being passed in. It should then return the calculated value. There is example code in the function that will show you how to do this. You will need to put in the real conversion formula.

The formula for converting Celsius to Fahrenheit is:

$$F = 9/5 * C + 32$$

You will also need to call the function from the main loop function. You will pass in the Celsius value you read in. The call to the function should look similar to this, but with your own variable names:

```
float myFVal = convertCelsiusToFahrenheit(myTempC);
```

After reading the temperature in from the sensor and converting it, the main loop prints out the temperature to the serial monitor.

Open the serial monitor (icon at top right) to see this output.

Figure 2: Serial monitor icon

Step 4: Extra Practice

Add some LEDs (don't forget the resistors!) to your board. Light up different LEDs based on the temperature. You could have different LEDs light up if the temperature is in different ranges, or you could have more LEDs light up the higher the temperature reading. Write your own functions to handle controlling the LEDs.

Appendix D: Survey

Pre and Post Questions

The following questions were given to both the Scratch and Arduino courses at both the beginning of Session 1 and the end of Session 5.

Survey ID: _____
Date: _____
Class Year: Freshman Sophomore Junior Senior
Class: Computer Apps Programming I Programming II Programming III AP Comp Science
Other _____
For the following questions, assume: x = 10 y = 5 z = 1
1) To what do the following expressions evaluate? a) $x + y$ _____ b) $x - y$ _____ c) $z + x + y$ _____
2) At the end of the following code, what is c equal to? a = 3 b = a - 1 a = b * 2 c is _____ c = a + b + 1
3) Which of the following conditionals evaluate to true and which to false? a) $x > y$ _____ b) x equals y _____ c) $(x > 9)$ and $(y > 10)$ _____ d) $(x < 0)$ or $(y < 0)$ or $(z > 0)$ _____
4) What does the following loop output? (Remember, y = 5) while (y < 12) { print "Hello " y = y + 2 } _____

5) Given the following functions, what does function3 output?

```
function1() {  
    print "My name "  
}  
  
function2(name) {  
    print "is " name  
}  
  
function3() {  
    call function1(), call function2("Annie")  
}
```

6) Please answer these questions as they apply to you today:

a) How comfortable are you with computers?

(On a scale from 1 – 5, 1 = not comfortable at all; 5 = very comfortable)

1 2 3 4 5

b) How many years programming experience have you had?

c) If so, with what platforms?

C/C++

Java

Scratch

Visual Basic

Arduino

Other _____

d) How likely would you be to choose a computing major in college? (Computer Science, Computer Engineering, Software Engineering, Information Science, etc.)

(On a scale from 1 – 5, 1 = no chance; 3 = would consider it; 5 = definitely will)

1 2 3 4 5

e) How well do you understand

(On a scale from 1 – 5, 1 = not at all; 3 = somewhat; 5 = very well)

1) variables	1	2	3	4	5
2) conditionals	1	2	3	4	5
3) loops	1	2	3	4	5
4) functions	1	2	3	4	5
5) computer programming	1	2	3	4	5

Scratch Post Questions

The following questions were given to the Scratch course at the end of Session 5.

f) How easy did you find learning Scratch?

(On a scale from 1 – 5, 1 = very difficult; 5 = very easy)

1 2 3 4 5

g) How interesting did you find learning Scratch?

(On a scale from 1 – 5, 1 = totally boring; 3 = ok; 5 = very interesting)

1 2 3 4 5

h) Did learning Scratch change your image of computing?

Yes – Positively Yes – Negatively No

i) Which labs did you complete?

(On a scale from 1 – 5, 1 = didn't do; 4 = finished; 5 = finished extra practice section, N/A = no attempt)

1) Lab 1 – Introduction

1 2 3 4 5 N/A

2) Lab 2 – Variables

1 2 3 4 5 N/A

3) Lab 3 – Conditionals / If Statements

1 2 3 4 5 N/A

4) Lab 4 – Iteration / Loops

1 2 3 4 5 N/A

5) Lab 5 – Functions

1 2 3 4 5 N/A

7) Was there anything particular that you liked about the lectures / labs?

8) What did you NOT like about lectures / labs?

Arduino Post Questions

The following questions were given to the Arduino course at the end of Session 5.

f) How easy did you find learning Arduino?

(On a scale from 1 – 5, 1 = very difficult; 5 = very easy)

1 2 3 4 5

g) How interesting did you find learning Arduino?

(On a scale from 1 – 5, 1 = totally boring; 3 = ok; 5 = very interesting)

1 2 3 4 5

h) Did learning Arduino change your image of computing?

Yes – Positively Yes – Negatively No

i) Which labs did you complete?

(On a scale from 1 – 5, 1 = didn't do; 4 = finished; 5 = finished extra practice section, N/A = no attempt)

1) Lab 1 – Introduction

1 2 3 4 5 N/A

2) Lab 2 – Variables

1 2 3 4 5 N/A

3) Lab 3 – Conditionals / If Statements

1 2 3 4 5 N/A

4) Lab 4 – Iteration / Loops

1 2 3 4 5 N/A

5) Lab 5 – Functions

1 2 3 4 5 N/A

7) Was there anything particular that you liked about the lectures / labs?

8) What did you NOT like about lectures / labs?

Appendix E: Detailed Results

In the below tables, we describe the results for each individual question. Listed is the question text and ratio of students receiving a given score to the total number of students taking both surveys. We also show the mean difference, standard deviation and mode of scores in the pre-survey and post-survey, as well as the same aggregate functions for the difference between the pre-survey and post-survey scores.

Objective Questions

Question 1a: Variables

For the following questions, assume: x = 10 y = 5 z = 1 To what do the following expressions evaluate? a) $x + y$ _____ b) $x - y$ _____				
Group 1 (Arduino)				
Score	% with score in pre-survey		% with score in post-survey	
1/1	85.71%		100.00%	
0/1	0.00%		0.00%	
Aggregates		Pre-survey	Post-survey	Difference
	Mean	2.57	3.00	0.43
	S.D.	1.05	0.00	1.05
	Mode	3	2	0

Question 1b: Variables

For the following questions, assume:

$$x = 10$$

$$y = 5$$

$$z = 1$$

To what do the following expressions evaluate?

a) $x + y$ _____

b) $x - y$ _____

c) $z + x + y$ _____

Group 2 (Scratch)

Score	% with score in pre-survey	% with score in post-survey
3/3	91.67%	91.67%
2/3	0.00%	0.00%
1/2	0.00%	0.00%
0/3	8.33%	8.33%

Aggregates		Pre-survey	Post-survey	Difference
	Mean	2.75	2.75	0.00
	S.D.	0.83	0.83	0.00
	Mode	3	3	0
Group 3 (Arduino)				
Score	% with score in pre-survey		% with score in post-survey	
3/3	100.00%		100.00%	
2/3	0.00%		0.00%	
1/2	0.00%		0.00%	
0/3	0.00%		0.00%	
Aggregates		Pre-survey	Post-survey	Difference
	Mean	3.00	3.00	0.00
	S.D.	0.00	0.00	0.00
	Mode	3	3	0
Group 4 (Scratch)				
Score	% with score in pre-survey		% with score in post-survey	
3/3	100.00%		87.50%	
2/3	0.00%		0.00%	
1/2	0.00%		0.00%	
0/3	0.00%		12.50%	
Aggregates		Pre-survey	Post-survey	Difference
	Mean	3.00	2.63	-0.38
	S.D.	0.00	0.99	0.99
	Mode	3	3	0
Group 5 (Scratch and Arduino)				
Score	% with score in pre-survey		% with score in post-survey	
3/3	100.00%		100.00%	
2/3	0.00%		0.00%	
1/2	0.00%		0.00%	
0/3	0.00%		0.00%	
Aggregates		Pre-survey	Post-survey	Difference
	Mean	3.00	3.00	0.00
	S.D.	0.00	0.00	0.00
	Mode	3	3	0
Group 5 (Scratch and Arduino)				
Score	% with score in pre-survey		% with score in post-survey	
3/3	98.08%		96.15%	
2/3	0.00%		0.00%	
1/2	0.00%		0.00%	
0/3	1.92%		3.85%	
Aggregates		Pre-survey	Post-survey	Difference
	Mean	2.90	2.90	0.00
	S.D.	0.54	0.54	0.55
	Mode	3.00	3.00	0.00
Groups 2 - 5				

Score	% with score in pre-survey	% with score in post-survey		
3/3	98.08%	96.15%		
2/3	0.00%	0.00%		
1/2	0.00%	0.00%		
0/3	1.92%	3.85%		
Aggregates		Pre-survey	Post-survey	Difference
	Mean	2.94	2.88	-0.06
	S.D.	0.41	0.58	0.41
	Mode	3.00	3.00	0.00

Question 2: Variables

At the end of the following code, what is c equal to? <pre> a = 3 b = a - 1 a = b * 2 c is _____ c = a + b + 1 </pre>		
Group 2 (Scratch)		
Score	% with score in pre-survey	% with score in post-survey
1/1	41.67%	41.67%
0/1	58.33%	58.33%

Aggregates		Pre-survey	Post-survey	Difference
	Mean	0.42	0.42	0.00
	S.D.	0.49	0.49	0.58
	Mode	0	0	0
Group 3 (Arduino)				
Score	% with score in pre-survey		% with score in post-survey	
1/1	84.62%		92.31%	
0/1	15.38%		7.69%	
Aggregates		Pre-survey	Post-survey	Difference
	Mean	0.85	0.92	0.08
	S.D.	0.36	0.27	0.27
	Mode	1	1	0
Group 4 (Scratch)				
Score	% with score in pre-survey		% with score in post-survey	
1/1	12.50%		37.50%	
0/1	87.50%		62.50%	
Aggregates		Pre-survey	Post-survey	Difference
	Mean	0.13	0.38	0.25
	S.D.	0.33	0.48	0.43
	Mode	0	0	0
Group 5 (Scratch and Arduino)				
Score	% with score in pre-survey		% with score in post-survey	
1/1	42.11%		63.16%	
0/1	57.89%		26.32%	
Aggregates		Pre-survey	Post-survey	Difference
	Mean	0.42	0.71	0.21
	S.D.	0.49	0.46	0.61
	Mode	0	1	0
All Groups				
Score	% with score in pre-survey		% with score in post-survey	
1/1	48.08%		61.54%	
0/1	51.92%		38.46%	
Aggregates		Pre-survey	Post-survey	Difference
	Mean	0.48	0.62	0.13
	S.D.	0.50	0.49	0.52
	Mode	0.00	1.00	0.00

Question 3: Conditionals

For the following questions, assume:

$x = 10$

$y = 5$

$z = 1$

Which of the following conditionals evaluate to true and which to false?

(&& means “and”, || means “or”, == means “equals”)

a) $x > y$ _____

b) $x == y$ _____

c) $(x > 9) \&\& (y > 10)$ _____

d) $(x < 0) || (y < 0) || (z > 0)$ _____

Group 1 (Arduino)

Score	% with score in pre-survey	% with score in post-survey
4/4	42.86%	28.57%
3/4	57.14%	71.43%
2/4	0.00%	0.00%
1/4	0.00%	0.00%
0/4	0.00%	0.00%

Aggregates		Pre-survey	Post-survey	Difference
	Mean	3.43	3.29	-0.14
	S.D.	0.49	0.45	0.64
	Mode	3	3	0
Group 2 (Scratch)				
Score	% with score in pre-survey		% with score in post-survey	
4/4	41.67%		58.33%	
3/4	33.33%		25.00%	
2/4	0.00%		8.33%	
1/4	0.00%		0.00%	
0/4	16.67%		8.33%	
Aggregates		Pre-survey	Post-survey	Difference
	Mean	2.67	3.25	0.58
	S.D.	1.60	1.16	1.44
	Mode	4	4	0
Group 3 (Arduino)				
Score	% with score in pre-survey		% with score in post-survey	
4/4	100.00%		92.31%	
3/4	0.00%		0.00%	
2/4	0.00%		0.00%	
1/4	0.00%		0.00%	
0/4	0.00%		7.69%	
Aggregates		Pre-survey	Post-survey	Difference
	Mean	4.00	3.69	-0.31
	S.D.	0.00	1.07	1.07
	Mode	4	4	0
Group 4 (Arduino)				
Score	% with score in pre-survey		% with score in post-survey	
4/4	25.00%		37.50%	
3/4	37.50%		25.00%	
2/4	25.00%		0.00%	
1/4	0.00%		0.00%	
0/4	12.50%		37.50%	
Aggregates		Pre-survey	Post-survey	Difference
	Mean	2.63	2.25	-0.38
	S.D.	1.22	1.79	1.58
	Mode	3	0	0
Group 5 (Scratch and Arduino)				
Score	% with score in pre-survey		% with score in post-survey	

	21.05%		26.32%	
3/3	68.42%		57.89%	
2/3	5.26%		5.26%	
1/2	0.00%		5.26%	
0/3	5.26%		5.26%	
Aggregates		Pre-survey	Post-survey	Difference
	Mean	3.00	2.95	-0.05
	S.D.	0.86	1.00	1.36
	Mode	3	3	0
All Groups				
Score	% with score in pre-survey		% with score in post-survey	
4/4	45.76%		49.15%	
3/4	40.68%		35.59%	
2/4	5.08%		3.39%	
1/4	0.00%		1.69%	
0/4	8.47%		10.17%	
Aggregates		Pre-survey	Post-survey	Difference
	Mean	3.15	3.12	-0.03
	S.D.	1.12	1.22	1.33
	Mode	4.00	4.00	0.00

Question 4: Iteration

What does the following loop output? (Remember, y = 5) <pre> while (y < 12) { print "Hello " y = y + 2 } </pre>				
Group 1 (Arduino)				
Score	% with score in pre-survey		% with score in post-survey	
1/1	14.29%		57.14%	
0/1	85.71%		42.86%	
Aggregates		Pre-survey	Post-survey	Difference
	Mean	0.14	0.57	0.43
	S.D.	0.35	0.49	0.49
	Mode	0	1	0
Group 2				
Score	% with Score in pre-survey		% with score in post-survey	
1/1	16.67%		8.33%	
0/1	83.33%		91.67%	

Aggregates		Pre-survey	Post-survey	Difference
	Mean	0.17	0.08	-0.08
	S.D.	0.37	0.28	0.28
	Mode	0	0	0
Group 3 (Arduino)				
Score	% with Score in pre-survey		% with score in post-survey	
1/1	69.23%		61.54%	
0/1	30.77%		38.46%	
Aggregates		Pre-survey	Post-survey	Difference
	Mean	0.69	0.62	-0.08
	S.D.	0.46	0.49	0.62
	Mode	1	1	0
Group 4 (Scratch)				
Score	% with Score in pre-survey		% with score in post-survey	
1/1	0.00%		0.00%	
0/1	100.00%		100.00%	
Aggregates		Pre-survey	Post-survey	Difference
	Mean	0.00	0.00	0.00
	S.D.	0.00	0.00	0.00
	Mode	0	0	0
Group 5 (Scratch and Arduino)				
Score	% with Score in pre-survey		% with score in post-survey	
1/1	0.00%		0.00%	
0/1	100.00%		100.00%	
Aggregates		Pre-survey	Post-survey	Difference
	Mean	0.00	0.00	0.00
	S.D.	0.00	0.00	0.00
	Mode	0	0	0
All Groups				
Score	% with Score in pre-survey		% with score in post-survey	
1/1	20.34%		22.03%	
0/1	79.66%		77.97%	
Aggregates		Pre-survey	Post-survey	Difference
	Mean	0.20	0.22	0.02
	S.D.	0.40	0.41	0.39
	Mode	0.00	0.00	0.00

Question 5: Functions

Given the following functions, what does function3 output?

```
function1() {
    print "My name "
}
function2(name) {
    print "is " name
}
function3() {
    call function1(), call function2("Annie")
}
```

Group 1 (Arduino)

Score	% with score in pre-survey	% with score in post-survey		
1/1	14.29%	42.86%		
0/1	85.71%	57.14%		
Aggregates		Pre-survey	Post-survey	Difference
	Mean	0.14	0.43	0.29
	S.D.	0.35	0.49	0.45
	Mode	0	0	0

Group 2 (Scratch)

Score	% with Score in pre-survey	% with score in post-survey
1/1	33.33%	58.33%
0/1	66.67%	41.67%

Aggregates		Pre-survey	Post-survey	Difference
	Mean	0.33	0.58	0.25
	S.D.	0.47	0.49	0.60
	Mode	0	1	0
Group 3 (Arduino)				
Score	% with Score in pre-survey		% with score in post-survey	
1/1	92.31%		92.31%	
0/1	7.69%		7.69%	
Aggregates		Pre-survey	Post-survey	Difference
	Mean	0.92	0.92	0.00
	S.D.	0.27	0.27	0.39
	Mode	1	1	0
Group 4 (Scratch)				
Score	% with Score in pre-survey		% with score in post-survey	
1/1	12.50%		37.50%	
0/1	87.50%		62.50%	
Aggregates		Pre-survey	Post-survey	Difference
	Mean	0.13	0.38	0.25
	S.D.	0.33	0.48	0.43
	Mode	0	0	0
Group 5 (Scratch and Arduino)				
Score	% with Score in pre-survey		% with score in post-survey	
1/1	36.84%		42.11%	
0/1	63.16%		57.89%	
Aggregates		Pre-survey	Post-survey	Difference
	Mean	0.37	0.42	0.05
	S.D.	0.48	0.49	0.69
	Mode	0	0	0
All Groups				
Score	% with Score in pre-survey		% with score in post-survey	
1/1	42.37%		57.63%	
0/1	57.63%		42.37%	
Aggregates		Pre-survey	Post-survey	Difference
	Mean	0.42	0.58	0.15
	S.D.	0.49	0.49	0.58
	Mode	0.00	1.00	0.00

Computing Experience

Question 6: Comfort with Computers

How comfortable are you with computers? (On a scale from 1 – 5, 1 = not comfortable at all; 5 = very comfortable)				
Group 1 (Arduino)				
Response	% with response in pre-survey		% with response in post-survey	
5	14.29%		57.14%	
4	57.14%		42.86%	
3	14.29%		0.00%	
2	0.00%		0.00%	
1	0.00%		0.00%	
Unknown	14.29%		0.00%	
Aggregates		Pre-survey	Post-survey	Difference
	Mean	4.00	4.57	0.50
	S.D.	0.58	0.49	0.50
	Mode	4	5	1
Group 2 (Scratch)				
Response	% with response in pre-survey		% with response in post-survey	
5	8.33%		8.33%	
4	0.00%		8.33%	
3	25.00%		25.00%	
2	0.00%		16.67%	
1	58.33%		33.33%	
Unknown	8.33%		8.33%	

Aggregates		Pre-survey	Post-survey	Difference
	Mean	1.91	2.36	0.44
	S.D.	1.31	1.30	0.68
	Mode	1	1	0
Group 3 (Arduino)				
Response	% with response in pre-survey		% with response in post-survey	
5	38.46%		38.46%	
4	23.08%		53.85%	
3	38.46%		0.00%	
2	0.00%		0.00%	
1	0.00%		0.00%	
Unknown	0.00%		7.69%	
Aggregates		Pre-survey	Post-survey	Difference
	Mean	4.00	4.42	0.33
	S.D.	0.88	0.49	0.62
	Mode	3	4	0
Group 4 (Scratch)				
Response	% with response in pre-survey		% with response in post-survey	
5	0.00%		0.00%	
4	0.00%		0.00%	
3	25.00%		37.50%	
2	25.00%		0.00%	
1	37.50%		37.50%	
Unknown				
Aggregates		Pre-survey	Post-survey	Difference
	Mean	1.86	2.00	0.00
	S.D.	0.83	1.00	0.58
	Mode	1	3	0
Group 5 (Scratch and Arduino)				
Response	% with response in pre-survey		% with response in post-survey	
5	21.05%		15.79%	
4	21.05%		15.79%	
3	36.84%		42.11%	
2	21.05%		26.32%	
1	0.00%		0.00%	
Unknown	0.00%		0.00%	
Aggregates		Pre-survey	Post-survey	Difference
	Mean	3.42	3.21	-0.21
	S.D.	1.04	1.00	0.77
	Mode	3	3	0

Question 9: Likelihood to Choose Computing Major

How likely would you be to choose a computing major in college? (Computer Science, Computer Engineering, Software Engineering, Information Science, etc.)

(On a scale from 1 – 5, 1 = no chance; 3 = would consider it; 5 = definitely will)

Group 1 (Arduino)		
Response	% with response in pre-survey	% with response in post-survey
5	28.57%	28.57%
4	0.00%	0.00%
3	42.86%	42.86%
2	0.00%	28.57%
1	28.57%	0.00%
Blank	0.00%	0.00%

Aggregates		Pre-survey	Post-survey	Difference
	Mean	3.00	3.29	0.29
	S.D.	1.51	1.16	0.45
	Mode	3	3	0
Group 2 (Scratch)				
Response	% with response in pre-survey		% with response in post-survey	
5	8.33%		8.33%	
4	0.00%		0.00%	
3	8.33%		8.33%	
2	33.33%		16.67%	
1	41.67%		58.33%	
Unknown	8.33%		8.33%	
Aggregates		Pre-survey	Post-survey	Difference
	Mean	1.91	1.73	-0.18
	S.D.	1.16	1.21	0.39
	Mode	1	1	0
Group 3 (Arduino)				
Response	% with response in pre-survey		% with response in post-survey	
5	38.46%		38.46%	
4	30.77%		7.69%	
3	30.77%		46.15%	
2	0.00%		0.00%	
1	0.00%		0.00%	
Unknown	0.00%		7.69%	
Aggregates		Pre-survey	Post-survey	Difference
	Mean	4.08	3.92	-0.25
	S.D.	0.83	0.95	0.72
	Mode	5	3	0
Group 4 (Scratch)				
Response	% with response in pre-survey		% with response in post-survey	
5	0.00%		0.00%	
4	0.00%		0.00%	
3	0.00%		0.00%	
2	12.50%		12.50%	
1	87.50%		87.50%	
Unknown	0.00%		0.00%	
Aggregates		Pre-survey	Post-survey	Difference
	Mean	1.13	1.13	0.00
	S.D.	0.33	0.33	0.00
	Mode	1	1	0
Group 5 (Scratch and Arduino)				
Response	% with response in pre-survey		% with response in post-survey	

5	5.26%	10.53%		
4	0.00%	0.00%		
3	26.32%	15.79%		
2	42.11%	26.32%		
1	26.32%	47.37%		
Unknown	0.00%	0.00%		
Aggregates		Pre-survey	Post-survey	Difference
	Mean	2.16	2.00	-0.16
	S.D.	0.99	1.26	0.81
	Mode	2	1	0

Question 10: Understanding Variables

How well do you understand variables? (On a scale from 1 – 5, 1 = not at all; 3 = somewhat; 5 = very well)				
Group 1 (Arduino)				
Response	% with response in pre-survey		% with response in post-survey	
5	0.00%		28.57%	
4	14.29%		42.86%	
3	71.43%		28.57%	
2	0.00%		0.00%	
1	14.29%		0.00%	
Unknown	0.00%		0.00%	
Aggregates		Pre-survey	Post-survey	Difference
	Mean	2.86	4.00	1.14
	S.D.	0.83	0.76	1.12
	Mode	3	4	0
Group 2 (Scratch)				
Response	% with response in pre-survey		% with response in post-survey	
5	16.67%		16.67%	
4	8.33%		16.67%	
3	33.33%		8.33%	
2	0.00%		25.00%	
1	33.33%		25.00%	
Unknown	8.33%		8.33%	

Aggregates		Pre-survey	Post-survey	Difference
	Mean	2.73	2.73	0.00
	S.D.	1.48	1.48	0.85
	Mode	1	1	0
Group 3 (Arduino)				
Response	% with response in pre-survey		% with response in post-survey	
5	53.85%		84.62%	
4	46.15%		7.69%	
3	0.00%		0.00%	
2	0.00%		0.00%	
1	0.00%		0.00%	
Unknown	0.00%		7.69%	
Aggregates		Pre-survey	Post-survey	Difference
	Mean	4.54	4.92	0.33
	S.D.	0.50	0.28	0.47
	Mode	5	5	0
Group 4 (Scratch)				
Response	% with response in pre-survey		% with response in post-survey	
5	0.00%		12.50%	
4	25.00%		12.50%	
3	25.00%		25.00%	
2	37.50%		25.00%	
1	0.00%		0.00%	
Unknown	12.50%		25.00%	
Aggregates		Pre-survey	Post-survey	Difference
	Mean	2.86	3.17	0.00
	S.D.	0.83	1.07	0.58
	Mode	2	3	0
Group 5 (Scratch)				
Response	% with response in pre-survey		% with response in post-survey	
5	31.58%		15.79%	
4	15.79%		5.26%	
3	10.53%		42.11%	
2	0.00%		15.79%	
1	42.11%		21.05%	
Unknown	0.00%		0.00%	
Aggregates		Pre-survey	Post-survey	Difference
	Mean	2.95	2.79	-0.16
	S.D.	1.76	1.28	1.60
	Mode	1	3	0
All Groups				
Response				

5	25.42%	32.20%		
4	22.03%	13.56%		
3	22.03%	22.03%		
2	5.08%	13.56%		
1	22.03%	11.86%		
Unknown	3.39%	6.78%		
Aggregates		Pre-survey	Post-survey	Difference
	Mean	3.25	3.44	0.20
	S.D.	1.48	1.41	1.20
	Mode	5.00	5.00	0.00

Question 11: Understanding Conditionals

How well do you understand conditionals? (On a scale from 1 – 5, 1 = not at all; 3 = somewhat; 5 = very well)				
Group 1 (Arduino)				
Response	% with response in pre-survey		% with response in post-survey	
5	0.00%		57.14%	
4	28.57%		14.29%	
3	42.86%		28.57%	
2	14.29%		0.00%	
1	14.29%		0.00%	
Unknown	0.00%		0.00%	
Aggregates		Pre-survey	Post-survey	Difference
	Mean	2.86	4.29	1.43
	S.D.	0.99	0.88	1.40
	Mode	3	5	3
Group 2 (Scratch)				
Response	% with response in pre-survey		% with response in post-survey	
5	8.33%		8.33%	
4	0.00%		25.00%	
3	8.33%		8.33%	
2	8.33%		25.00%	
1	66.67%		25.00%	
Unknown	8.33%		8.33%	

Aggregates		Pre-survey	Post-survey	Difference
	Mean	1.64	2.64	1.00
	S.D.	1.23	1.37	0.95
	Mode	1	4	0
Group 3 (Arduino)				
Response	% with response in pre-survey		% with response in post-survey	
5	15.38%		38.46%	
4	69.23%		46.15%	
3	7.69%		7.69%	
2	0.00%		0.00%	
1	7.69%		0.00%	
Unknown	0.00%		7.69%	
Aggregates		Pre-survey	Post-survey	Difference
	Mean	3.85	4.33	0.15
	S.D.	0.95	0.62	1.35
	Mode	4	4	0
Group 4 (Scratch)				
Response	% with response in pre-survey		% with response in post-survey	
5	0.00%		12.50%	
4	0.00%		12.50%	
3	25.00%		12.50%	
2	37.50%		12.50%	
1	25.00%		25.00%	
Unknown	12.50%		25.00%	
Aggregates		Pre-survey	Post-survey	Difference
	Mean	2.00	2.67	0.00
	S.D.	0.76	1.49	0.58
	Mode	2	1	0
Group 5 (Scratch and Arduino)				
Response	% with response in pre-survey		% with response in post-survey	
5	0.00%		0.00%	
4	0.00%		10.53%	
3	15.79%		26.32%	
2	10.53%		31.58%	
1	73.68%		31.58%	
Unknown	0.00%		0.00%	
Aggregates		Pre-survey	Post-survey	Difference
	Mean	1.42	2.16	0.74
	S.D.	0.75	0.99	0.91
	Mode	1	2	0
All Groups				
Response	% with response in pre-survey		% with response in post-survey	

5	5.08%	18.64%		
4	18.64%	22.03%		
3	16.95%	16.95%		
2	11.86%	16.95%		
1	44.07%	18.64%		
Unknown	3.39%	6.78%		
Aggregates		Pre-survey	Post-survey	Difference
	Mean	2.26	3.05	0.79
	S.D.	1.34	1.42	1.03
	Mode	1	4	0

Question 12: Understanding Iteration

How well do you understand loops? (On a scale from 1 – 5, 1 = not at all; 3 = somewhat; 5 = very well)				
Group 1 (Arduino)				
Response	% with response in pre-survey		% with response in post-survey	
5	0.00%		42.86%	
4	14.29%		28.57%	
3	42.86%		28.57%	
2	28.57%		0.00%	
1	14.29%		0.00%	
Unknown	0.00%		0.00%	
Aggregates		Pre-survey	Post-survey	Difference
	Mean	2.57	4.14	1.29
	S.D.	0.90	0.83	0.70
	Mode	3	5	1
Group 2 (Scratch)				
Response	% with response in pre-survey		% with response in post-survey	
5	8.33%		8.33%	
4	0.00%		25.00%	
3	16.67%		8.33%	
2	0.00%		16.67%	
1	66.67%		33.33%	
Unknown	8.33%		8.33%	

Aggregates		Pre-survey	Post-survey	Difference
	Mean	1.73	2.55	0.82
	S.D.	1.29	1.44	1.11
	Mode	1	1	0
Group 3 (Arduino)				
Response	% with response in pre-survey		% with response in post-survey	
5	15.38%		15.38%	
4	61.54%		53.85%	
3	15.38%		23.08%	
2	7.69%		0.00%	
1	0.00%		0.00%	
Unknown	0.00%		7.69%	
Aggregates		Pre-survey	Post-survey	Difference
	Mean	3.85	3.92	-0.23
	S.D.	0.77	0.64	0.80
	Mode	4	4	0
Group 4 (Scratch)				
Response	% with response in pre-survey		% with response in post-survey	
5	0.00%		0.00%	
4	0.00%		12.50%	
3	12.50%		25.00%	
2	25.00%		0.00%	
1	50.00%		37.50%	
Unknown	12.50%		25.00%	
Aggregates		Pre-survey	Post-survey	Difference
	Mean	1.57	2.17	0.00
	S.D.	0.73	1.21	0.58
	Mode	1	1	0
Group 5 (Scratch and Arduino)				
Response	% with response in pre-survey		% with response in post-survey	
5	5.26%		5.26%	
4	5.26%		15.79%	
3	21.05%		36.84%	
2	21.05%		21.05%	
1	47.37%		21.05%	
Unknown	0.00%		0.00%	
Aggregates		Pre-survey	Post-survey	Difference
	Mean	1.47	3.00	1.53
	S.D.	0.94	1.30	1.09
	Mode	1	3	2
All Groups				
Response	% with response in pre-survey		% with response in post-survey	

5	6.78%	15.25%		
4	15.25%	27.12%		
3	13.56%	27.12%		
2	16.95%	5.08%		
1	44.07%	18.64%		
Unknown	3.39%	6.78%		
Aggregates		Pre-survey	Post-survey	Difference
	Mean	2.21	3.16	0.93
	S.D.	1.35	1.33	1.22
	Mode	1	4	0

Question 13: Understanding Functions

How well do you understand functions? (On a scale from 1 – 5, 1 = not at all; 3 = somewhat; 5 = very well)				
Group 1 (Arduino)				
Response	% with response in pre-survey		% with response in post-survey	
5	6.78%		15.25%	
4	15.25%		27.12%	
3	13.56%		27.12%	
2	16.95%		5.08%	
1	44.07%		18.64%	
Unknown	3.39%		6.78%	
Aggregates		Pre-survey	Post-survey	Difference
	Mean	3.14	4.14	1.00
	S.D.	0.64	0.83	1.07
	Mode	3	5	0
Group 2 (Scratch)				
Response	% with response in pre-survey		% with response in post-survey	
5	8.33%		8.33%	
4	8.33%		25.00%	
3	16.67%		16.67%	
2	16.67%		8.33%	
1	41.67%		33.33%	
Unknown	8.33%		8.33%	

Aggregates		Pre-survey	Post-survey	Difference
	Mean	2.18	2.64	0.45
	S.D.	1.34	1.43	0.66
	Mode	1	1	0
Group 3 (Arduino)				
Response	% with response in pre-survey		% with response in post-survey	
5	7.69%		30.77%	
4	30.77%		38.46%	
3	38.46%		15.38%	
2	7.69%		7.69%	
1	15.38%		0.00%	
Unknown	0.00%		7.69%	
Aggregates		Pre-survey	Post-survey	Difference
	Mean	3.08	4.00	0.62
	S.D.	1.14	0.91	1.86
	Mode	3	4	0
Group 4 (Scratch)				
Response	% with response in pre-survey		% with response in post-survey	
5	0.00%		12.50%	
4	12.50%		12.50%	
3	37.50%		12.50%	
2	25.00%		12.50%	
1	12.50%		25.00%	
Unknown	12.50%		25.00%	
Aggregates		Pre-survey	Post-survey	Difference
	Mean	2.57	2.67	0.50
	S.D.	0.90	1.49	1.12
	Mode	3	1	0
Group 5 (Scratch and Arduino)				
Response	% with response in pre-survey		% with response in post-survey	
5	5.26%		5.26%	
4	5.26%		15.79%	
3	21.05%		36.84%	
2	21.05%		21.05%	
1	47.37%		21.05%	
Unknown	0.00%		0.00%	
Aggregates		Pre-survey	Post-survey	Difference
	Mean	2.00	2.63	0.63
	S.D.	1.17	1.13	1.11
	Mode	1	3	1
All Groups				
Response	% with response in pre-survey		% with response in post-survey	

5	5.08%	16.95%		
4	15.25%	23.73%		
3	30.51%	23.73%		
2	16.95%	11.86%		
1	28.81%	16.95%		
Unknown	3.39%	6.78%		
Aggregates		Pre-survey	Post-survey	Difference
	Mean	2.49	3.13	0.65
	S.D.	1.22	1.35	1.13
	Mode	3	4	0

Question 14: Understanding Computer Programming

How well do you understand computer programming? (On a scale from 1 – 5, 1 = not at all; 3 = somewhat; 5 = very well)				
Group 1 (Arduino)				
Response	% with response in pre-survey		% with response in post-survey	
5	0.00%		28.57%	
4	42.86%		71.43%	
3	42.86%		0.00%	
2	14.29%		0.00%	
1	0.00%		0.00%	
Unknown	0.00%		0.00%	
Aggregates		Pre-survey	Post-survey	Difference
	Mean	3.29	4.29	1.00
	S.D.	0.70	0.45	0.53
	Mode	4	4	1
Group 2 (Scratch)				
Response	% with response in pre-survey		% with response in post-survey	
5	8.33%		8.33%	
4	0.00%		8.33%	
3	25.00%		25.00%	
2	0.00%		16.67%	
1	58.33%		33.33%	
Unknown	8.33%		8.33%	

Aggregates		Pre-survey	Post-survey	Difference
	Mean	1.91	2.36	0.44
	S.D.	1.31	1.30	0.68
	Mode	1	1	0
Group 3 (Arduino)				
Response	% with response in pre-survey		% with response in post-survey	
5	0.00%		7.69%	
4	53.85%		23.08%	
3	30.77%		53.85%	
2	15.38%		7.69%	
1	0.00%		0.00%	
Unknown	0.00%		7.69%	
Aggregates		Pre-survey	Post-survey	Difference
	Mean	3.38	3.33	-0.31
	S.D.	0.74	0.75	1.07
	Mode	4	3	0
Group 4 (Scratch)				
Response	% with response in pre-survey		% with response in post-survey	
5	0.00%		0.00%	
4	0.00%		0.00%	
3	25.00%		37.50%	
2	25.00%		0.00%	
1	37.50%		37.50%	
Unknown				
Aggregates		Pre-survey	Post-survey	Difference
	Mean	1.86	2.00	0.00
	S.D.	0.83	1.00	0.58
	Mode	1	3	0
Group 5 (Scratch and Arduino)				
Response	% with response in pre-survey		% with response in post-survey	
5	0.00%		0.00%	
4	0.00%		0.00%	
3	10.53%		47.37%	
2	21.05%		26.32%	
1	68.42%		26.32%	
Unknown	0.00%		0.00%	
Aggregates		Pre-survey	Post-survey	Difference
	Mean	1.42	2.21	0.94
	S.D.	0.67	0.83	0.75
	Mode	1	3	1
All Groups				
Response	% with response in pre-survey		% with response in post-survey	

5	1.69%	6.78%		
4	16.95%	15.25%		
3	23.73%	37.29%		
2	15.25%	13.56%		
1	38.98%	20.34%		
Unknown	3.39%	6.78%		
Aggregates		Pre-survey	Post-survey	Difference
	Mean	2.25	2.73	0.47
	S.D.	1.20	1.18	0.81
	Mode	1	3	0

Experience with Labs

Question 15: Scratch easiness

How easy did you find learning Scratch? (On a scale from 1 – 5, 1 = very difficult; 5 = very easy)		
Group 2 (Scratch)		
Response	% with response in pre-survey	
5	8.33%	
4	25.00%	
3	25.00%	
2	16.67%	
1	16.67%	
Unknown	8.33%	
Aggregates	Mean	2.91
	S.D.	1.24
	Mode	3

Group 4 (Scratch)		
Response	% with response in pre-survey	
5	12.50%	
4	25.00%	
3	37.50%	
2	25.00%	
1	0.00%	
Unknown	0.00%	
Aggregates	Mean	3.25
	S.D.	0.97
	Mode	3
Group 5 (Scratch and Arduino)		
Response	% with response in pre-survey	
5	31.58%	
4	42.11%	
3	21.05%	
2	0.00%	
1	5.26%	
Unknown	0.00%	
Aggregates	Mean	3.95
	S.D.	1.00
	Mode	4

Question 16: Scratch Interestingness

How interesting did you find learning Scratch? (On a scale from 1 – 5, 1 = totally boring; 3 = ok; 5 = very interesting)		
Group 2 (Scratch)		
Response	% with response in pre-survey	
5	16.67%	
4	0.00%	
3	25.00%	
2	25.00%	
1	25.00%	
Unknown	8.33%	
Aggregates	Mean	2.55
	S.D.	1.37
	Mode	3

Group 4 (Scratch)		
Response	% with response in pre-survey	
5	12.50%	
4	37.50%	
3	12.50%	
2	25.00%	
1	12.50%	
Unknown	0.00%	
Aggregates	Mean	3.13
	S.D.	1.27
	Mode	4
Group 5 (Scratch and Arduino)		
Response	% with response in pre-survey	
5	36.84%	
4	15.79%	
3	21.05%	
2	26.32%	
1	0.00%	
Unknown	0.00%	
Aggregates	Mean	3.63
	S.D.	1.22
	Mode	5

Question 17: Arduino Easiness

How easy did you find learning Arduino? (On a scale from 1 – 5, 1 = very difficult; 5 = very easy)		
Group 1 (Arduino)		
Response	% with response in pre-survey	
5	14.29%	
4	28.57%	
3	57.14%	
2	0.00%	
1	0.00%	
Unknown	0.00%	
Aggregates	Mean	3.57
	S.D.	0.73
	Mode	3

Group 3 (Arduino)		
Response	% with response in pre-survey	
5	7.69%	
4	30.77%	
3	38.46%	
2	7.69%	
1	7.69%	
Unknown	7.69%	
Aggregates	Mean	3.25
	S.D.	1.01
	Mode	3
Group 5 (Scratch and Arduino)		
Response	% with response in pre-survey	
5	5.26%	
4	15.79%	
3	31.58%	
2	21.05%	
1	26.32%	
Unknown	0.00%	
Aggregates	Mean	2.53
	S.D.	1.19
	Mode	3

Question 18: Arduino Interestingness

How interesting did you find learning Arduino? (On a scale from 1 – 5, 1 = totally boring; 3 = ok; 5 = very interesting)		
Group 1 (Arduino)		
Response	% with response in pre-survey	
5	28.57%	
4	57.14%	
3	14.29%	
2	0.00%	
1	0.00%	
Unknown	0.00%	
Aggregates	Mean	4.14
	S.D.	0.64
	Mode	4

Group 3 (Arduino)		
Response	% with response in pre-survey	
5	15.38%	
4	30.77%	
3	46.15%	
2	0.00%	
1	0.00%	
Unknown	7.69%	
Aggregates	Mean	3.67
	S.D.	0.75
	Mode	3
Group 5 (Scratch and Arduino)		
Response	% with response in pre-survey	
5	10.53%	
4	26.32%	
3	26.32%	
2	15.79%	
1	21.05%	
Unknown	0.00%	
Aggregates	Mean	2.89
	S.D.	1.29
	Mode	3

Image of Computing

Question 19: Change in Image of Computing - Scratch

Did learning Scratch change your image of computing?	
Yes – Positively Yes – Negatively No	
Group 2 (Scratch)	
Response	% with response in pre-survey
Yes - Positively	16.67%
Yes - Negatively	25.00%
No	50.00%
Unknown	8.33%
Group 4 (Scratch)	
Response	% with response in pre-survey
Yes - Positively	37.50%
Yes - Negatively	12.50%
No	43.75%
Unknown	6.25%
Group 5 (Scratch and Arduino)	
Response	% with response in pre-survey
Yes - Positively	52.63%

Yes - Negatively	0.00%
No	36.84%
Unknown	10.53%

Question 20: Change in Image of Computing - Arduino

Did learning the Arduino platform change your image of computing? (On a scale from 1 – 5, 1 = not at all; 3 = somewhat; 5 = completely)		
Group 1 (Arduino)		
Response	% with response in post-survey	
5	28.57%	
4	42.86%	
3	28.57%	
2	0.00%	
1	0.00%	
Unknown	0.00%	
Aggregates	Mean	4.00
	S.D.	0.76
	Mode	4

Labs Completed

Group 1 (Arduino)

Which labs did you complete? (On a scale from 1 – 5, 1 = didn't do; 4 = finished; 5 = finished extra practice section, N/A = no attempt – if you were absent)		
Group 1 (Arduino)		
Lab 1 – Blinking Light		
Response	% with response in pre-survey	
5	85.71%	
4	14.29%	
3	0.00%	
2	0.00%	
1	0.00%	
N/A	0.00%	
Unknown	0.00%	
Aggregates	Mean	4.86
	S.D.	0.35
	Mode	5

Lab 2 – Dimmer Switch		
Response	% with response in pre-survey	
5	28.57%	
4	57.14%	
3	14.29%	
2	0.00%	
1	0.00%	
N/A	0.00%	
Unknown	0.00%	
Aggregates	Mean	4.14
	S.D.	0.64
	Mode	4
Lab 3a – Traffic Light		
Response	% with response in pre-survey	
5	42.86%	
4	28.57%	
3	14.29%	
2	0.00%	
1	14.29%	
N/A	0.00%	
Unknown	0.00%	
Aggregates	Mean	3.86
	S.D.	1.36
	Mode	5
Lab 4 – Piezo Buzzer		
Response	% with response in pre-survey	
5	28.57%	
4	28.57%	
3	14.29%	
2	0.00%	
1	28.57%	
N/A	0.00%	
Unknown	0.00%	
Aggregates	Mean	3.29
	S.D.	1.58
	Mode	4
Lab 5 – Temperature Sensor		
Response	% with response in pre-survey	
5	14.29%	
4	71.43%	
3	14.29%	
2	0.00%	
1	0.00%	
N/A	0.00%	

Unknown	0.00%	
Aggregates	Mean	4.00
	S.D.	0.53
	Mode	4

Group 2 (Scratch)

Which labs did you complete? (On a scale from 1 – 5, 1 = didn't do; 4 = finished; 5 = finished extra practice section, N/A = no attempt – if you were absent)		
Lab 1 – Moving Sprites		
Response	% with response in pre-survey	
5	16.67%	
4	33.33%	
3	16.67%	
2	16.67%	
1	0.00%	
N/A	8.33%	
Unknown	8.33%	
Aggregates	Mean	3.60
	S.D.	1.02
	Mode	4

Lab 2 – MadLibs		
Response	% with response in pre-survey	
5	16.67%	
4	25.00%	
3	16.67%	
2	25.00%	
1	0.00%	
N/A	8.33%	
Unknown	8.33%	
Aggregates	Mean	3.40
	S.D.	1.11
	Mode	2
Lab 3 – Tag Game		
Response	% with response in pre-survey	
5	16.67%	
4	33.33%	
3	8.33%	
2	25.00%	
1	0.00%	
N/A	8.33%	
Unknown	8.33%	
Aggregates	Mean	3.50
	S.D.	1.12
	Mode	4
Lab 4 – Sound		
Response	% with response in pre-survey	
5	16.67%	
4	16.67%	
3	16.67%	
2	25.00%	
1	0.00%	
N/A	8.33%	
Unknown	16.67%	
Aggregates	Mean	3.33
	S.D.	1.15
	Mode	2
Lab 5 – Calculator		
Response	% with response in pre-survey	
5	16.67%	
4	25.00%	
3	16.67%	
2	16.67%	
1	8.33%	
N/A	8.33%	

Unknown	8.33%	
Aggregates	Mean	3.30
	S.D.	1.27
	Mode	4

Group 3 (Arduino)

Which labs did you complete? (On a scale from 1 – 5, 1 = didn't do; 4 = finished; 5 = finished extra practice section, N/A = no attempt – if you were absent)		
Lab 1 – Blinking Light		
Response	% with response in pre-survey	
5	53.85%	
4	30.77%	
3	7.69%	
2	0.00%	
1	0.00%	
N/A	0.00%	
Unknown	7.69%	
Aggregates	Mean	4.50
	S.D.	0.65
	Mode	5

Lab 2 – Dimmer Switch		
Response	% with response in pre-survey	
5	30.77%	
4	38.46%	
3	23.08%	
2	0.00%	
1	0.00%	
N/A	0.00%	
Unknown	7.69%	
Aggregates	Mean	4.08
	S.D.	0.76
	Mode	4
Lab 3 – Light Button Switch		
Response	% with response in pre-survey	
5	15.38%	
4	30.77%	
3	38.46%	
2	7.69%	
1	0.00%	
N/A	0.00%	
Unknown	7.69%	
Aggregates	Mean	3.58
	S.D.	0.86
	Mode	3
Lab 4 – Piezo Buzzer		
Response	% with response in pre-survey	
5	15.38%	
4	53.85%	
3	15.38%	
2	7.69%	
1	0.00%	
N/A	0.00%	
Unknown	7.69%	
Aggregates	Mean	3.83
	S.D.	0.80
	Mode	4
Lab 5 – Temperature Sensor		
Response	% with response in pre-survey	
5	23.08%	
4	23.08%	
3	23.08%	
2	7.69%	
1	15.38%	
N/A	0.00%	

Unknown	7.69%	
Aggregates	Mean	3.33
	S.D.	1.37
	Mode	5

Group 4 (Scratch)

Which labs did you complete? (On a scale from 1 – 5, 1 = didn't do; 4 = finished; 5 = finished extra practice section, N/A = no attempt – if you were absent)		
Lab 1 – Moving Sprites		
Response	% with response in pre-survey	
5	0.00%	
4	12.50%	
3	50.00%	
2	0.00%	
1	0.00%	
N/A	12.50%	
Unknown	25.00%	
Aggregates	Mean	3.20
	S.D.	0.40
	Mode	3

Lab 2 – MadLibs		
Response	% with response in pre-survey	
5	0.00%	
4	12.50%	
3	50.00%	
2	0.00%	
1	0.00%	
N/A	12.50%	
Unknown	25.00%	
Aggregates	Mean	3.20
	S.D.	0.40
	Mode	3
Lab 3 – Tag Game		
Response	% with response in pre-survey	
5	0.00%	
4	25.00%	
3	25.00%	
2	12.50%	
1	0.00%	
N/A	12.50%	
Unknown	25.00%	
Aggregates	Mean	3.20
	S.D.	0.75
	Mode	4
Lab 4 – Sound		
Response	% with response in pre-survey	
5	0.00%	
4	37.50%	
3	25.00%	
2	0.00%	
1	0.00%	
N/A	12.50%	
Unknown	25.00%	
Aggregates	Mean	3.60
	S.D.	0.49
	Mode	4
Lab 5 – Calculator		
Response	% with response in pre-survey	
5	0.00%	
4	62.50%	
3	0.00%	
2	0.00%	
1	0.00%	
N/A	12.50%	

Unknown	25.00%	
Aggregates	Mean	4.00
	S.D.	0.00
	Mode	4

Group 5 (Scratch and Arduino)

Which labs did you complete? (On a scale from 1 – 5, 1 = didn't do; 4 = finished; 5 = finished extra practice section, N/A = no attempt – if you were absent)		
Lab 1 – Scratch / MadLibs		
Response	% with response in pre-survey	
5	47.37%	
4	21.05%	
3	10.53%	
2	15.79%	
1	5.26%	
N/A	0.00%	
Unknown	0.00%	
Aggregates	Mean	3.89
	S.D.	1.33
	Mode	5

Lab 2 – Scratch / Tag Game		
Response	% with response in pre-survey	
5	26.32%	
4	31.58%	
3	15.79%	
2	5.26%	
1	10.53%	
N/A	10.53%	
Unknown	0.00%	
Aggregates	Mean	3.65
	S.D.	1.34
	Mode	4
Lab 3 – Scratch / Sound		
Response	% with response in pre-survey	
5	47.37%	
4	36.84%	
3	5.26%	
2	10.53%	
1	0.00%	
N/A	0.00%	
Unknown	0.00%	
Aggregates	Mean	4.21
	S.D.	0.97
	Mode	5
Lab 4 – Arduino / Blinking light		
Response	% with response in pre-survey	
5	42.11%	
4	31.58%	
3	5.26%	
2	5.26%	
1	10.53%	
N/A	5.26%	
Unknown	0.00%	
Aggregates	Mean	3.94
	S.D.	1.38
	Mode	5
Lab 5 – Arduino / Dimmer switch		
Response	% with response in pre-survey	
5	21.05%	
4	31.58%	
3	15.79%	
2	0.00%	
1	10.53%	
N/A	21.05%	

Unknown	0.00%	
Aggregates	Mean	3.67
	S.D.	1.26
	Mode	4
Lab 6 – Arduino / Piezo Buzzer		
Response	% with response in pre-survey	
5	36.84%	
4	47.37%	
3	5.26%	
2	5.26%	
1	5.26%	
N/A	0.00%	
Unknown	0.00%	
Aggregates	Mean	4.05
	S.D.	0.99
	Mode	4

Question 31 – 32: Likes and Dislikes

Likes	Dislikes
Group 1 (Arduino)	
I enjoyed actually seeing what goes into hardware.	The lectures were dense.
Yes, I like the buzzer thing.	No, I like everything.
They let you choose the type of program language to use.	Too long.
Went more in-depth by converting our code in to something physical.	There wasn't much except maybe a little more instruction on write-ups.
I liked the blinking light lab and the sound lab, also the Scratch.	Not really, they were all pretty fun but lab 3 was difficult at times but it was still pretty fun.
It was fun learning how to do the Arduino	No not really
I liked working with Scratch and Arduinos. It was an interesting experience.	Not really.
Group 2 (Scratch)	
Play w/ Scratch	Difficult to understand
Nothing	They're boring.
Nothing	Boring
Nope.	Pretty boring, didn't learn much
No	Nothing
Interesting	Sometimes confusing
I liked Scratch.	Not anything.
The sounds in Scratch :D	I was confused! :(
I liked learning how to make a sprite.	I didn't know where to go for certain labs or where to save.
No	Don't know

Group 3 (Arduino)	
It was fun. The write-ups are easy to follow and helpful.	Libraries did not always work flawlessly.
Seeing the similarity to Java	Nothing
Easy to learn	No formal explanation of structure
They are fun and the Arduino is so adorable	Not very much time to do and then play with
The Arduinos are adorable!!! So cute! I hadn't used breadboards before & that was really fun!!	They were kind of hard.
It was new, interesting	Hard to understand, little frustrating
Group 4 (Scratch)	
	It's hard
It was fun, easy to understand, very clean and simple.	
Moving the sprites	
Annie is nice :)	Be louder & get control
Scratch was entertaining	
Not really / I liked moving the sprites	I couldn't follow, going to fast over lessons.
Not really, I liked watching the cat do different stuff	nothing, it was fine
Group 5 (Scratch and Arduino)	
I liked how in Scratch you could make a game and play it right there.	Sometimes the lectures got really confusing.
No they were straightforward.	The way (or) format of the programs
It was fun being able to control something.	Arduino was really hard with all the code.
What I liked about Scratch was making your own game.	Arduino a little harder than Scratch.
No	Arduino was just a little confusing.
It was fun messing around in Scratch	Not really?
	They went too fast
The overall subject matter was intriguing, being that I'm completely new to it all	it's very particular
I liked making characters in Scratch	Didn't understand what you were talking about
Once we figured out how to do it, the actual programming was fun.	Sometimes things were unexplained or confusing.
Scratch was fun to make games	It was alright
I liked being able to experiment with Arduinos	There was nothing I didn't like
They challenged me.	We had a lot of faulty equipment and I didn't learn much.
No	I did not understand the teacher. She was very confusing.
I just think it is interesting and I	

want to learn more	
Yes being able to control stuff	How confusing it was
No	How difficult it was. I would not consider programming as a profession.
Scratch was somewhat interesting, but I don't even know why you even teach Arduino, it is the most useless thing ever!	Scratch, not that much; Arduino, everything

Appendix F: Computing Careers Supplemental Survey

The following questions were given to Group 5 and a supplemental Computer Applications class at Morro Bay High School to assess their understanding of computing fields. The descriptions were taken from ACM / IEEE computing curricula in Computer Science, Computer Engineering, Software Engineering, Information Technology and Information Systems [35].

Computing is a broad field that can be split up in to the below disciplines.

First, please review the definitions of the following five disciplines. **Then**, categorize each activity as something that would be done by someone in one of the disciplines. Write the two-letter acronym (i.e. CS for Computer Science) in the blank next to the activity.

Computing Disciplines

1. **Computer Science (CS)** – "Computer science spans a wide range, from its theoretical and algorithmic foundations to cutting-edge developments in robotics, computer vision, intelligent systems, bioinformatics, and other exciting areas. They devise new ways to use computers. They design and implement software. They develop effective ways to solve computing problems."
2. **Computer Engineering (CE)** – "Computer engineering is concerned with the design and construction of computers and computer-based systems. It involves the study of hardware, software, communications, and the interaction among them."
3. **Software Engineering (SE)** – "Software engineering is the discipline of developing and maintaining software systems that behave reliably and efficiently, are affordable to develop and maintain, and satisfy all the requirements that customers have defined for them."
4. **Information Technology (IT)** – "Information Technology focuses on preparing graduates who are concerned with issues related to advocating for users and meeting their needs within an organizational and societal context through the selection, creation, application, integration and administration of computing technologies."
5. **Information Systems (IS)** – "Information systems specialists focus on integrating information technology solutions and business processes to meet the information needs of businesses and other enterprises, enabling them to achieve their objectives in an effective, efficient way."

Activities

1. Design and write a new programming language
2. Create software for computer graphics used in movies
3. Design an algorithm to map a person's genes
4. Work on a robot to vacuum a house
5. Design new computer chips to control x-ray machines
6. Write software to control satellites
7. Work with a team to design software for a specific client
8. Test code that other people have written
9. Gather requirements for marketing software
10. Install wireless network routers in an office

11. Administer a company's email server
12. Help a school install educational software
13. Work with a finance department to select accounting software
14. Negotiate a contract with a web hosting company
15. Manage a team of people working at a help desk

Results

Group 5

Participants	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	% Correct
Key	CS	CS	CS	CE	CE	CE	SE	SE	SE	IT	IT	IT	IS	IS	IS	100%
1	CE	IS	IT	CS	CE	SE	SE	IS	IS	CS	CE	SE	IT	IS	IT	20%
2	SE	CE	IT	IS	SE	CS	CE	IS	IT	IT	IS	CE	SE	IT	IS	13%
3	SE	CS	SE	CE	SE	SE	IS	IT	IS	CE	IS	SE	IT	IS	IT	20%
4	IS	IT	IS	CS	CE	SE	CS	IT	IS	CS	IT	CS	IT	CE	IT	13%
5	IS	IS	CS	CS	CS	SE	IT	SE	IT	CS	IT	SE	CE	IS	CE	27%
6	CS	SE	IT	CS	IT	SE	CS	IT	SE	IS	IT	SE	SE	IS	IS	33%
6	SE	SE	CS	CE	IT	SE	IS	IT	CE	IT	CS	IT	IS	CS	CE	33%
7	CS	SE	CS	CS	SE	IT	CE	CS	IS	SE	IT	SE	IT	CE	IS	27%
8	SE	SE	CS	IT	IS	CS	CE	IS	IT	CS	CE	CE	SE	CS	CE	7%
8	SE	SE	CS	CE	CS	SE	IT	SE	IS	IT	IS	IT	IT	IS	IS	47%
9	CS	SE	IT	CS	CE	SE	IT	SE	IS	IT	IS	IT	IT	IS	CE	40%
10	SE	SE	CS	CS	CS	CE	SE	CE	IS	IT	IS	IT	IS	IS	IS	53%
11	IS	SE	CS	CE	IT	SE	SE	CS	IT	IS	CS	CS	SE	IT	IS	27%
11	SE	SE	IT	CE	IS	SE	SE	CE	IT	IS	SE	SE	CE	IT	IS	20%
12	CS	CE	CS	CS	CS	CE	CS	SE	IT	CE	SE	SE	IT	SE	IS	33%
13	CE	IT	SE	CS	IS	CS	SE	IT	IS	IS	IS	SE	IS	CS	CE	13%
14	CE	CS	SE	CS	IS	CE	SE	SE	IT	IS	CE	CS	SE	IT	CE	27%
14	SE	CS	SE	CE	IS	SE	IT	IT	IS	IS	SE	SE	SE	CS	IS	20%
16	SE	SE	CS	CS	CS	CE	CE	IT	IS	CE	SE	IT	IS	SE	CE	27%
17	CS	CE	IT	SE	IS	SE	CS	SE	IT	CS	IS	CS	IS	IT	SE	20%
20	SE	SE	IT	CE	CE	SE	SE	IS	IT	CS	IT	IT	SE	CS	IS	40%
21	CS	SE	CS	CE	CE	SE	SE	SE	IS	IT	IT	CE	IS	SE	IS	67%
22	IS	IS	CS	CS	CE	SE	SE	IT	IT	CE	IT	SE	IS	IS	CE	40%
22	CS	CE	CS	CS	CE	CS	IT	CS	IS	CE	IS	CE	SE	IS	IT	27%
% CS	33%	17%	58%	58%	21%	17%	17%	13%	0%	25%	8%	17%	0%	21%	0%	
% CE	13%	17%	0%	38%	38%	21%	17%	8%	4%	21%	13%	17%	8%	8%	33%	
% SE	46%	54%	17%	4%	13%	67%	46%	33%	8%	4%	17%	46%	33%	13%	4%	
% IT	0%	8%	29%	4%	13%	4%	21%	33%	42%	33%	33%	29%	29%	21%	17%	
% IS	17%	13%	4%	4%	25%	0%	8%	17%	54%	25%	33%	0%	33%	46%	54%	

Group 6

Participants	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	% Correct
Key	CS	CS	CS	CE	CE	CE	SE	SE	SE	IT	IT	IT	IS	IS	IS	100%
1	CE	CS	IS	CS	CE	SE	IS	IT	IT	IS	IS	SE	SE	IS	IT	20%
2	CE	SE	IS	CS	CE	SE	SE	CE	SE	IS	IT	SE	SE	IT	CS	27%
3	CS	CE	IT	CE	SE	IS	IS	IT	IT	CE	SE	SE	CS	IT	IT	13%
4	CE	CS	CS	IT	CS	SE	IT	IS	IT	IS	IS	IT	IT	SE	IS	27%
5	SE	CE	IT	IS	CS	IS	CE	SE	IS	CS	CE	CE	IT	CS	IS	13%
6	IT	CS	IS	SE	CE	SE	SE	CS	IT	SE	IS	SE	SE	CS	IS	27%
7	CE	CS	IS	CS	IS	SE	SE	IS	CE	SE	IS	SE	SE	CS	IS	20%
8	CE	SE	CS	CS	CE	CS	IS	SE	SE	CE	IS	CS	IT	IS	CS	33%
9	CE	CS	CS	SE	SE	SE	SE	CE	IS	SE	IS	IS	SE	IS	IT	27%
10	CE	SE	CE	CS	CE	SE	CE	CS	SE	CE	IS	IT	SE	IS	IS	33%
11	CE	CE	CE	SE	CE	IT	SE	IT	IS	SE	IS	CS	SE	CS	IS	20%
12	SE	SE	CS	CS	CS	SE	IT	IS	IT	IT	IT	IS	IS	IS	CS	40%
13	CS	SE	IT	CS	CE	SE	SE	IS	IT	CE	IS	CE	SE	IS	IT	27%
14	CE	CE	CS	CS	SE	CS	IT	IS	IT	IS	IT	IS	IS	IS	IT	27%
% CS	14%	36%	36%	57%	21%	14%	0%	14%	0%	7%	0%	14%	7%	29%	21%	
% CE	64%	29%	14%	7%	50%	0%	14%	14%	7%	29%	7%	14%	0%	0%	0%	
% SE	14%	36%	0%	21%	21%	64%	50%	21%	29%	29%	7%	36%	57%	7%	0%	
% IT	7%	0%	21%	7%	0%	7%	21%	21%	50%	14%	29%	29%	21%	14%	36%	
% IS	0%	0%	29%	7%	7%	14%	21%	36%	21%	29%	64%	14%	21%	57%	50%	