ASSESSING COMPUTATIONAL THINKING IN COMPUTER SCIENCE UNPLUGGED ACTIVITIES

by

Brandon R. Rodriguez

A thesis submitted to the Faculty and the Board of Trustees of the Colorado School of Mines in partial fulfillment of the requirements for the degree of Master of Science (Computer Science).

Golden, Colorado

Date _____

Signed: _____
Brandon R. Rodriguez

Signed: _____
Dr. Tracy K. Camp
Thesis Advisor

Signed: _____
Dr. Cyndi Rader
Thesis Advisor

Golden, Colorado

Date _____

Signed: _____
Dr. Atef Elsherbeni
Professor and Head
Department of Electrical Engineering
and Computer Science

ABSTRACT

There is very little research on assessing computational thinking without using a programming language, despite the wide adoption of activities that teach these concepts without a computer, such as CS Unplugged. Measuring student achievement using CS Unplugged is further complicated by the fact that most activities are kinesthetic and team-oriented, which contrasts traditional assessment strategies designed for lectures and individual tasks. To address these issues, we have created an assessment strategy that uses a combination of in-class assignments and a final project. The assessments are designed to test different computational thinking principles, and use a variety of problem structures. The assessments were evaluated using a well-defined rubric along with notes from classroom observations to discover the extent CS Unplugged activities promote computational thinking. The results from our experiment include several statistically significant shifts supporting the hypothesis that students are learning computational thinking skills from CS Unplugged. Student performance across all of the worksheets gave insight into where problems can be improved or refined such that a greater number of students can reach proficiency in the subject areas.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

I want to thank Drs. Tracy Camp and Cyndi Rader, my advisors, for their tireless support throughout my journey. I am forever grateful for their patience as I learned the intricacies of doing research involving computer science, education, and human subjects. Both have been amazing mentors throughout my time at Colorado School of Mines. I also want to thank Terry Bridgman and Dr. Christopher Painter-Wakefield for serving on my thesis committee and being flexible as I filed the endless paperwork and scheduled the numerous meetings that a Master's degree entails. Lastly, I want to thank my parents, grandparents, and siblings for always standing behind me, and all of my friends and family for encouraging me to pursue my Master's degree.

None of my research would have been possible without the funding and support from the National Science Foundation under grants CNS-1240964 and DGE-0801692. Any opinions, findings, conclusions, or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

CHAPTER 1

INTRODUCTION

In 2006, Jeannette Wing (Carnegie Mellon University) coined the term *Computational Thinking* (CT) as a way that humans conceptualize computable problems [1]. Computational thinking is the "how" in problem solving, and is useful in answering unstructured problems or interpreting and understanding data. Computational thinking is important when there are many solutions that can lead to a correct answer, and where some solutions may offer a computational advantage when using a machine to calculate the result. Since computers are pervasive in our society, teaching CT concepts to more than university CS majors will give students the tools for effectively solving a variety of problems in different disciplinary areas.

The Computer Science Teachers Association (CSTA) has further refined the broad ideas of CT into five categories: data representation, decomposition, abstraction, algorithmic thinking, and patterns [2]. Data representation is the ability to take a type of data (images, text, sounds, etc.) and represent the information in a fashion that may initially be unintuitive, but useful for processing by a computer. An example of data representation is taking an image and using RGB components to characterize each color instead of using descriptive strings of text ("blue" versus R:0 G:0 B:255). Decomposition is breaking a problem into smaller pieces. Decomposition can often be used to separate a seemingly complex task into many simple tasks in order to solve the original problem. Abstraction deals with generalizing a problem to see if techniques from similar problems can be used to solve the current task at hand. Algorithmic thinking is designing step-by-step processes and applying known algorithms to obtain a solution. Finally, pattern recognition allows students to identify trends or discover the cause of the patterns.

During the 2000s, there was a sharp decrease in enrollment and interest in computer science at the university level [3]. As a result, many efforts and research projects aimed to

make programming more accessible. Two common examples are MIT's Scratch, and CMU's Alice visual programming languages [4–6]. The use of these visual languages helps students avoid details of contemporary programming languages such as semicolons and strict syntax rules. Research studies have shown that students are able to learn to program in these visual languages, but few papers have effectively mapped programming languages to computational thinking concepts.

Computational thinking encompasses much more than learning how to program. Computer Science Unplugged (CS Unplugged) activities are a set of lesson plans made available for free on the internet. The aim of these lesson plans is twofold: they act as a way to convey fundamental computer science concepts to students without any computer skills, and they work to bridge the gap between K-12 teachers who may not have a technical background but are expected to teach technical ideas. CS Unplugged activities are kinesthetic, engaging, and above all do not require students to know a programming language or have access to a computer. If the energy and enthusiasm for CS Unplugged activities can be successfully combined with the educational components of computational thinking, then we will have an effective means to empower students to solve problems.

1.1 Background

CS Unplugged activities have been shown to be effective and engaging in teaching the same concepts as alternative, traditional methods [7, 8]. Through a partnership with STEM School and Academy (STEM School), our project has been successful in pilot testing numerous CS Unplugged activities here in Colorado. Extensions have been developed for some original CS Unplugged activities to make them appropriate and challenging for a middle school environment. Additionally, new CS Unplugged activities have been created and pilot tested to offer a variety of topics for sixth and seventh grade students. To help students make connections between CS concepts and their daily lives, the new activities and extensions have focused on creating real-world links.

Original CS Unplugged activities were deployed in after-school and outreach settings (and not classroom environments). To address the lack of learning objectives, lesson plans were developed and tested to ensure that a full 60-minute period could be filled with any single CS Unplugged activity. Two teachers working at STEM School have worked with our team to generate valuable observation feedback and contributed to new extensions and assessment ideas. Students also provided feedback on the activities through a post-deployment survey administered anonymously via Google Docs. Feedback from both the teachers and students was used to evaluate the design of new activities and remove any material that was too confusing or advanced for middle school students.

To assess the impact on student attitudes, pre- and post-surveys were deployed. The pre- and post- deployment student survey results have shown an increase in computing career knowledge (for all pilot deployments), and confidence in computing (for some pilots). In other words, initial results show promise for CS Unplugged activities, but do not demonstrate any growth in student *learning*. The assessment of student learning, specifically to computational thinking, is the primary focus of this research.

1.2 Research Goals

Very little research has been done on the ability of CS Unplugged activities to teach computational thinking. By targeting middle school students in the seventh grade with an assessment that combines the kinesthetic components of CS Unplugged and the ideas of CT, we hope to answer the following questions:

- Can we develop an effective instrument to determine what CT principles students are acquiring from the kinesthetic CS Unplugged activities?
  - What approaches can we incorporate from evidence-centered design assessment?
  - What ideas can we employ from other CT assessments?
- Do CS Unplugged activities encourage computational thinking?

By answering these questions, we can better evaluate the contribution of CS Unplugged activities as a vehicle for learning CT skills.

CHAPTER 2

RELATED WORK

There are many projects underway to develop lesson plans and assessments that can easily be introduced into an existing tech-ed classroom. In a study from John Carroll University, for example, researchers developed a learning progression model for elementary-aged students [9]. The study used a rubric to classify computer programs written by students in order to measure the sophistication of a program. Looking for the existence of particular code blocks in Scratch, or noting the use of more complicated constructs, allowed researchers to extract when (at what grade level) students began using these elements. For example, the authors assigned a weighted scale for students' use of conditionals. A simple "if" statement received a score of 1, an "if-else" statement received a 2, and a nested "if/if" statement or "if-elseif-..." statement received a 3. This research project straddles assessing CT (although authors did not make a formal link between code blocks and the CT principles) and teaching computer science using an introductory programming language.

Relevant research for assessing CT in CS Unplugged activities falls into three main areas. The first relates to CS Unplugged and related approaches for teaching the CT skill set. The second area deals with different models for assessing engagement and CT patterns. The third area involves evidence centered design (ECD) and the process of creating assessments to follow this ideology. While a number of studies have been performed in the areas of CT and CS Unplugged, there are almost no studies that address the intersection of these two topics with regards to assessments.

2.1 CS Unplugged and Related Approaches

Numerous research projects have presented methods for teaching computer science without the use of computers or programming languages. CS Unplugged and "computer science magic shows" have been successful examples of teaching computer science through highly

engaging activities for students [10–12]. CS Unplugged activities have been adopted by a variety of educational outreach programs such as after-school workshops and summer camps, and have even being incorporated into the Exploring CS Curriculum [13, 14]. The majority of the studies conducted using these types of activities have been concerned primarily with increasing interest in computer science, and not necessarily about incorporating the lesson plans into classroom environments or assessing what the students are learning by completing the activities.

Renate Thies and Jan Vahrenhold from the Technical University of Dortmund in Germany investigated the suitability of CS Unplugged activities for use in a classroom (instead of an after-school program) by teaching a group of students. They used CS Unplugged activities to teach half the students, and used alternative tools for the other half of the students. Their findings showed CS Unplugged activities were equally effective in transferring knowledge as there was no significant difference in achievement between the group who learned with CS Unplugged activities and the group who learned with alternative materials [7]. Additionally, the researchers studied the impact of using CS Unplugged activities in different grade levels, and found that the activities had a significant positive impact when used with middle school classes. Thies and Vahrenhold have also mapped CS Unplugged lessons to Bloom's Taxonomy to determine what level of cognitive processes are prompted by various activities [15].

Bloom's Revised Taxonomy consists of six levels of cognitive processes, each representing a different level of intellectual achievement [16]. The six levels, in order from basic to most complex, are: remembering, understanding, applying, analyzing, evaluating, and creating. Remembering is a tool often stressed in K-12 environments and utilized as a means of testing students. As the lowest level of Bloom's Taxonomy, "remembering" tasks are designed to test the ability to recall information presented to students. "Creating," on the other hand, asks students to design or develop a new product or point of view based on material previously presented to them. Creating is a much more difficult task, and thus a better indicator of

student comprehension of a subject than remembering or understanding information.

Thies and Vahrenhold's research examined all unmodified CS Unplugged activities. These activities were originally designed to be used in outreach scenarios, and therefore do not explicitly list learning objectives. The researchers' extrapolation of learning objectives suggests that the CS Unplugged curriculum lies in the lower end of the Bloom's Taxonomy spectrum. The authors noted that higher level learning objectives are needed for middle school audiences [7]. Thies and Vahrenhold's research is of particular significance because they bridge the gap between entertaining outreach programs and measurable student outcomes that can be used in a traditional classroom. The extensions and new activities our group has developed specifically address higher learning objectives in order to make CS Unplugged materials better suited for secondary education.

Two other significant studies involve general approaches to computing education. Quintin Cutts of the University of Glasgow detailed how group exercises in classroom environments can be just as effective as one-on-one tutors. Group work can also increase confidence and encourage students to become personally interested in the material [17]. Karen Brennan and Mitchel Resnick of MIT questioned how to get students to focus on what is learned (and supported) by computational thinking that isn't conveyed via existing coursework. One example they used was creating an animation in Scratch versus making a video using special editing software [4]. Both studies suggest that active participation in a lesson is helpful in shifting the perspectives of students.

Lynn Lambert of Christopher Newport University published an article in 2009 that deployed pre- and post-surveys to evaluate CS Unplugged activities [8]. Her survey included similar questions to those administered in our deployment. Lambert's results found students showed an increase in confidence in computing topics, but failed to gain knowledge about computing careers. The conclusions from Lambert's study were the basis for the development of career related extensions and lecture material for CS Unplugged activities, which is discussed in detail in the Approach section of this thesis.

Finally, the Bebras International Contest on Informatics and Computer Fluency (Bebras, for short), is a set of computation related questions that are unaffiliated with CS Unplugged activities [18]. Bebras challenges embody several components of computational thinking, and can be used without a computer. Bebras helped inspire some facets of the final project presented in this thesis for assessing CT in middle school students. Other studies have shown support for assessments that focus on understanding computational processes and CT skills as opposed to focusing on programming languages [19].

2.2 Educational Assessment Approaches

AgentSheets, a simulation environment, aimed to put the power of computing into the hands of everyday computer users in 1996, ten years before CT research began to gain momentum. The goal of AgentSheets is to provide the power to process and visualize data to people who had never taken a formal computer science course. The program creates Java applets to facilitate sharing of these simulations online, and includes an "Agent Exchange" where users are able to share pieces of their programs for easy reuse. One observation made by researchers was the need to incorporate interactivity into AgentSheets to increase engagement among users [20]. Perhaps the most intriguing takeaway, however, was that its user base ranged from elementary school students to high school students to doctors. The users of AgentSheets have to model their data, and are able to explore the effects of how changing parameters in a simulation can drastically change the overall outcome, which incorporates elements of computational thinking. The authors of AgentSheets developed a "Computational Thinking Pattern Quiz" that asked students eight questions on real-world video clips. Each video mimicked a CT pattern demonstrated in a "Frogger" style game [21, 22]. The results of the quiz showed that most of the participants were able to recognize and understand the thinking patterns in the real-world videos. In many ways, AgentSheets began answering questions about computer science education for the general public before many in the CS field began to ask the questions.

The analysis of programs written by students as a measure of CT has been criticized by a number of researchers. A study conducted at Stanford University noted that programs cannot be the only tool used to evaluate a student. The researchers argued that you cannot derive the student's thought process when they wrote the program, and found that students often cannot explain how or why their code works [6]. Their study looked at both Scratch and Alice as two sources of programs for assessment. Researchers augmented the evaluation of a student's learning with quizzes on CT terms and paper assessments where students paste Scratch blocks into place without being able to press "Play" and see the output. This approach provided better insight into the study results.

The above studies offer promising glimpses into the area of CT assessment, although none can be a direct model for our exclusively unplugged curriculum. Another study performed at Stanford does take a non-programming approach by interviewing students individually about an algorithmic efficiency problem [10]. Students (seventh graders) first brainstormed solutions out loud and explained their reasoning to the teacher. The teacher then provided three different solutions to the student before asking which of the three proposed would be best. The two-fold nature of this assessment makes it an interesting idea for seeing the process of a student's solution, and also being able to gauge the student's comprehension of computational thinking. Six students were interviewed individually for approximately 25 minutes after school or during their lunch period. Although the interviews yield great insights, the time investment to do individual student interviews makes it unrealistic to deploy at a large scale in a middle school classroom.

The Santa Fe Institute published an article on computational thinking in the K-8 curriculum, which suggested using a scaffolded final project. Unlike a traditional school assignment with a clearly stated rubric, the approach utilized in this article required students to complete a series of independent tasks that gradually became more complex. The idea was for students to be confident in their basic understanding before allowing more open-ended creativity towards the latter half of the project. Programs used in this project were Scratch and

a StarLogo ecosystem simulation to provide students with a real-world link to monitoring animal habitats. The core idea behind their scaffolding project was the "Use-Modify-Create" learning progression where students complete the first two steps via structured mini-activities before being given some creative control [5, 23].

2.3 Evidence Centered Design

SRI International (SRI) and the Educational Testing Service (ETS) are two organizations that have contributed to assessment research. The focus of this thesis is to apply established assessment techniques to CS Unplugged activities, not to define a new assessment process. One assessment paradigm promoted by both SRI and ETS is Evidence-Centered Design, or ECD. ECD assessments answer the questions "What skills should be assessed?" and "What student performances reveal those skills?" ECD arrives at an answer to these questions by using assessments as evidence to support what concepts students do and do not know or knowledge students do and do not have [24, 25].

SRI has a grant titled "Principled Assessment of Computational Thinking," or PACT, which has been active since 2012 [25]. The goal of their grant is to design, develop, and validate assessments for computational thinking by using evidence-centered design. Their grant has been awarded $690,000 to investigate the issue; this funding illustrates that the problem of developing a new assessment approach is not a trivial task. ECD can be broken down into five smaller assessment issues: domain analysis, domain modeling, conceptual assessment framework, assessment implementation, and assessment delivery.

The final project used in this thesis incorporates aspects of ECD. Our research team identified important CT areas in the CS Unplugged activities that should be tested, a first step in the domain analysis. Developed rubrics provided a framework to judge the results of the final project, and we pilot tested the project as part of our implementation and delivery of the new assessment tool.

CHAPTER 3
APPROACH

This section presents the approach used to answer our research questions, including a brief history of work done on the project from spring 2014 through spring 2015. Prior to assessing the acquisition of content knowledge, we wanted to ensure that the activities were engaging and appropriate for middle school students. The work completed in spring 2014, fall 2014, and spring 2015 is briefly summarized and grouped according to the school involved at that time, and major revisions to CS Unplugged activities are highlighted in these sections. A description of every activity used in the research is provided, including a list of the pilot tests when each activity was used. Then, a brief summary of the pre- and post- attitude surveys and results is presented. Finally, the proposed assessment strategy is detailed. The assessments make use of CS Unplugged extensions, as well as a modular final project.

3.1 Compass Montessori School (Compass)

Thies and Vahrenhold noted that the CS Unplugged activities, unmodified, were not sufficiently challenging to be used in a middle school classroom [7]. In response, the Mines research team developed extensions during the spring semester of 2014 to accompany several existing activities. Some of these extensions (career extensions) aimed to make stronger connections between the Unplugged activities and computing careers, while others (content extensions) were part of an ongoing effort to make CS Unplugged appropriate for students in grades 6-8, based on Bloom's Taxonomy (Bloom's). In the latter half of the spring 2014 semester, a pilot test of five activities and their associated extensions was conducted in combined 7th-9th grade classrooms at Compass Montessori School in Golden. A Mines graduate student presented the activities, while the teachers watched and collected their feedback in observation reports, which were used to drive subsequent revisions and edits.

Anonymous pre- and post-surveys were utilized in the Compass deployment to determine the impact of the CS Unplugged activities on students' interest and confidence in computing.

3.2 STEM School & Academy (STEM School)

Pilot tests continued in the fall of 2014 at STEM School in Highlands Ranch. Four deployments were completed throughout the course of the 2014-2015 school year with two teachers: one with 6th grade students and one with 7th grade students both semesters. Each deployment consisted of four class periods, and approximately 120 students total. The set of activities used in each pilot test varied among deployments and is documented in Table 3.1. Teachers provided observation reports and attended retrospective meetings at the end of each pilot test to collaborate on activity revisions.

Students were given modified pre- and post-surveys to collect their input. The post-survey also asked for students' favorite and least favorite activities. The tallied results of their favorite activities are presented in Table 3.2. The two lists are rough inverses of each other, validating the overall trend. The low-ranked activities were addressed and either revised or replaced with new activities created at Mines during the fall of 2014. The following three pilot tests continued this editing process until feedback was generally positive on all of the activities. A brief description of the revisions to and the origins of each activity, along with a table that maps the activities to computational thinking attributes, are included as Appendix A. This appendix also includes a table of the extensions used in low-rated activities and describes the possible causes of the ratings, as well as revisions made to address those concerns.

Table 3.1: A brief description of each activity used in our project, and its associated pilot test(s). The columns signify the semester ("F" for fall, "S" for spring), year (2014 or 2015) and grade level (6, 7, or 7-9) for each deployment.

		Deployment Semester & Grade Level				
	Activity Description	S14 7-9	F14 6	F14 7	S15 6	S15 7

Table 3.1: Continued

Artificial Intelligence	Students participate in a mock Turing Test and discuss intelligent agents.	X	X	X	X	X
Binary Numbers	Students learn binary/decimal conversion, binary addition, and how overflow occurs.	X	X		X	X
Cryptology and Information Hiding	Students share information without being identified, and use math and ciphers to catch a bank robber.		X			
Caesar Cipher & Frequency Analysis	Students explore Caesar and substitution ciphers, how the ciphers can be cracked, and why security is important.					X
Computer Vision	Students develop a better understanding of how computers "see" and the problems computer vision is solving.			X	X	
Deadlock and Client/Server Routing	Students participate in a deadlock group exercise before simulating a client/server image download.			X	X	X
Finite State Automata	Students model states and transitions in demonstrative examples before modeling FSAs for real-world objects.	X	X	X	X	
Image Representation	Students represent black and white images in binary and explore why compression is important.	X	X			
Minimal Spanning Trees	Students interact with graphs and find a least-cost solution to visit all nodes.	X	X		X	
Parity & Error Correction	Students learn about error detection and correction using 1D and 2D parity schemes.					X

Table 3.1: Continued

Sorting and Searching	Students cover four algorithms: linear search, binary search, selection sort, and quicksort, and apply these algorithms to different problems.		X			X

Table 3.2: Student activity data from the first STEM pilot test. The middle column reports activities marked as students' favorite (1 being the activity with the most votes, 7 being the activity with the least votes). The rightmost column reports activities marked as students' least favorite (1 being the activity with the most votes, 7 being the activity with the least votes).

Student Rankings		
	Favorite (1 = Most, 7 = Least)	Least Favorite (1 = Least, 7 = Most)
Finite State Automata	1	6
Binary Numbers	2	4
Artificial Intelligence	3	7
Minimal Spanning Trees	4	5
Sorting / Searching	5	1
Cryptology / Info Hiding	6	3
Image Representation	7	2

3.3 Student Attitudes About Computing

Students in CS Unplugged pilot tests (spring of 2014, fall of 2014 and spring of 2015 semesters) were asked to complete anonymous surveys about their computing perceptions. A copy of the questions asked in the surveys is included as Appendix B of this proposal. The surveys used a series of Likert scale questions grouped together to measure computing interest, confidence, outcome expectations, career knowledge, and students' intent to persist in CS. Additionally, the surveys asked students to identify their gender, ethnicity, and whether or not they had previously attended a "Discovering Technology" workshop at Colorado School of Mines. The post-survey also asked students to select their favorite and least favorite CS Unplugged activities from a list and to explain why. The survey was piloted and

revised during the spring 2014 and fall 2014 semesters, eventually reaching a stable state in spring 2015.

Surveys were collected electronically via a private Google Docs form (students were not able to view the results). Names of students were collected only to match the pre- and post-surveys after the classroom deployment; once ID numbers were assigned to each student, names were deleted. Students who were absent when either survey was administered were removed from the dataset. There were approximately 200 paired responses for the two deployments. The matched data was then anonymized and evaluated using a principal components analysis for the different survey question groupings. Initial analysis of the first dataset shows a statistically significant increase in computing confidence and outcome expectations, suggesting the classroom deployments are having an impact on student self-efficacy in computing.

Detailed analysis of the attitude surveys is outside the scope of this thesis, but initial results show that CS Unplugged activities are a promising approach to improve students' confidence, outcome expectations, and knowledge of computing careers. Before a strong case can be made to deploy CS Unplugged more widely, however, we need to have some idea of what students are learning via these activities. Answering that question is the primary objective of this thesis.

3.4 Assessments

As stated in the previous sections, content extensions were developed for a number of the CS Unplugged activities used in our project. The extensions were created to involve thinking at a higher level of Bloom's Taxonomy and, therefore, be more appropriate for use with middle school students. The higher levels of Bloom's Taxonomy focus on the ability to transfer knowledge to new problems, evaluate solutions, and create new points of view. Our assessment approach relies on worksheets related to five activities and a cumulative final project. Note that activity extensions are completed during the class period when material is taught. The final project builds on several activities and is administered in a separate

period where no new information is presented.

3.4.1 Binary Numbers

In the binary numbers lesson, students learn how to represent decimal numbers in binary format (and vice versa). The class answers questions as a group regarding the largest value that can be represented using one, two, three, and four bits. A worksheet, titled "Check Your Understanding" (see Appendix D), asks students the largest value that can be represented with five bits. Students can calculate the answer by adding up all the place values (1 + 2 + 4 + 8 + 16), or by noticing that the largest number represented by m bits is one less than the next place value ($2 * n - 1$, where $n = 2^{m-1}$). Thus, if a student knows the fifth place value is 16 ($n = 2^{5-1} = 16$), he or she can easily determine the maximum value five bits can hold is $2 * 16 - 1 = 31$.

The worksheet disguises a similar question by reversing the wording. "How many bits would you need to represent 63?" is the final question on the worksheet. Again, students can calculate the answer through trial and error, converting the decimal number to binary, or by using the same technique used to solve the prior question. Continuing the pattern from the previous paragraph, the sixth bit represents 32, so the maximum value a six bit number can hold is 32*2 - 1 = 63. These types of questions highlight the desired thought processes involved in computational thinking. While solvable multiple ways, the anticipated method is for the students to recognize the pattern and generalize the solution so they can apply it to all of the questions on the worksheet. In one pilot test, 75% of students correctly answered the five-bit question, but only 40% correctly answered the six-bit question. Results from the data collection deployment are presented in the results chapter.

3.4.2 Caesar Cipher and Frequency Analysis

The cryptology activity also includes several exercises with strong CT links. In this activity, students are introduced to the Caesar cipher before learning about substitution ciphers. Methods for breaking both of these encryption schemes are also presented in class.

Students are then given an in-class worksheet where they use a Caesar cipher for encrypting messages. The worksheet contains elements of data representation and abstraction, as students are representing plaintext with cipher text, and students must think about the Caesar cipher wheel and determine how many different keys exist. The second worksheet students complete relates to decrypting a message encoded with a substitution cipher. Students are not given the cipher, and must apply frequency analysis techniques to try and obtain the plaintext result. The substitution cipher worksheet requires students to use problem decomposition and pattern recognition as they must a) create a letter frequency table, and b) use the table to decrypt the message.

3.4.3 Minimal Spanning Trees

The minimal spanning tree activity is centered around working in pairs to produce a spanning tree across a city connected by roads. Students learn Kruskal's algorithm for finding a minimal spanning tree and practice finding a tree on a graph. The worksheet is laminated and used with place markers so the students can fail and iterate at a quicker speed than if they were required to erase pencil marks repeatedly. During the class, students also briefly touch on other types of graph problems, including the Chinese Postman problem and an intractable sets problem. The minimal spanning tree, however, consumes the majority of the class period and is the main focus of this lesson.

3.4.4 Parity and Error Detection

Parity and Error Detection is broken into two segments. The first segment, which takes approximately half of the class time, is all about parity in ASCII and one-dimension. Students add a new word to their vocabulary and practice detecting an error. Then, as a segue to the two-dimensional material, a "magic" trick is performed using 2D parity. Before the magic trick is explained, students learn about 2D parity and expand on the knowledge learned from the first half of the class.

3.4.5 Sorting and Searching

The revised sorting and searching CS Unplugged activity involves whole class demonstrations using ping pong balls and scales to illustrate different approaches to sorting and searching. Specifically, students reason about linear and binary searching, and selection and quicksort sorting algorithms. After the demonstrations, students are given two worksheets: one that deals with searching and one that deals with sorting. The exercises do not specify how the students should find the target in the searching task, or what method to use for ordering the objects in the sorting task. The worksheets are structured such that they can be reviewed at a later time to determine what algorithm (if any) the students used. The worksheets focus on the CT components of abstraction and algorithmic thinking, as the students were given four algorithms earlier in the class and are now being asked to transfer that knowledge to a new problem.

There are two separate activities that relate to sorting and searching. The first worksheet, which consists of numerous cows with various numbers printed on their sides, relates to searching. The cows pertain to a backstory to make the activity more interesting for students, and are irrelevant for completing the task. In the searching worksheet, there are two blocks of cows with numbers (this is a partner activity): one set of cows is sorted and the other set is not. In the first iteration of this activity, students are able to utilize a binary searching algorithm to find a specific cow, since the numbers are sorted. Students are asked to mark the worksheets so that we can reflect on their work later. We can observe whether or not students used binary search based on which cows they inquired about from their partner.

The other worksheet, which pertains to sorting, is slightly more straightforward. We can reflect on the students' work later based on the questions they asked while they were sorting colors. This worksheet is also a partner activity; while one student is sorting the colors, their partner has a mapping of colors to weights. Thus, every student should end up with the same answer (i.e., students do not create their own weights for the colors).

3.4.6 Finite State Automata

The finite state automata (FSA) lesson is primarily based around teams of four or so students. One person in each team is designated a fruit vendor who sells apples and bananas. The job of the other three students is to understand the pattern of how the vendor is selling the fruit; all vendors are given an instruction card, so their behaviors are identical. First, students try this without knowing how to represent an FSA. Then, students take turns telling different pieces of the pattern to the entire class as FSA conventions are being introduced. Finally, students apply what they've learned about FSAs in two worksheets to try and model a traffic light and fill in transitions in a treasure hunt map. These worksheets effectively capture the steps students take to solving the problem, which makes them ideal for review and scoring at a later date.

3.4.7 Final Project

Based on work presented in [5, 10], it has been shown that final projects and individual student interviews are effective methods to evaluate student understanding. The designed final project for use with this thesis has tried to capture the allure of CS Unplugged activities with the modular design of a scaffolded final project, while also incorporating open-ended questions so as not to directly lead students to a desired outcome. The final assessment combines some ideas from the Bebras contest with questions that require prior information covered in CS Unplugged activities [18]. The final project is expected to help measure knowledge retention and assess underlying CT concepts by presenting new problems not discussed in any CS Unplugged activity. Both forms of CT assessment (activity extensions and the final project) will be used to help verify and reinforce any findings. There are two versions of the final project with similar activities. This section describes one version in detail.

One assessment, named "Carnytown Carnival Murder Mystery," challenges students to apply concepts covered in various CS Unplugged activities to help solve the murder of a

carnival employee. The project is organized with five suspects that each have their own associated task for students to complete. These tasks are independent of one another, so an incorrect answer on one problem will not cascade into incorrect answers in other tasks. Upon correct completion of each task, the associated carnival employee will "give" the student a clue (i.e., the clue will need to be provided by the teacher). The clues provided by the five suspects can be combined in two different ways in order to produce the name of a top suspect.

An initial run-through of the final project with two undergraduate students familiar with the modified CS Unplugged activities showed that the project is cohesive. The undergraduates had not seen the assessment beforehand, but were still able to transfer knowledge from various activities and apply the concepts to new problems without any clarifications. The questions of the final project were later modified to produce a second, nearly identical project, but with different stories. This modified version of the final project is referred to as the "Pet" version because its problems have an animal theme.

The final project covers all five CT principles (data representation, decomposition, pattern recognition, abstraction, and algorithmic thinking). The assessment also reaches into the higher levels of Bloom's Taxonomy, making it age appropriate for middle school students and providing another pivot to evaluate the results. The first part of the project (named after the "Sammy" carnival character) relates to the "Binary Numbers" CS Unplugged activity. The worksheet serves to remind students about representing letters as binary numbers, and falls under the "remembering" and "understanding" classifications (the lowest levels) of Bloom's Taxonomy. "Tammy" builds on the graph concepts taught in the minimal spanning tree activity, has students using the algorithm on a new problem, and falls under the "applying" level of Bloom's Taxonomy. "Larry" uses concepts from the finite state automata activity. Students construct a FSA based on a paragraph of information, which involves pattern recognition, and falls under the "applying" and "analyzing" levels of Bloom's Taxonomy. "Barry" asks students to select and justify the most efficient solution to a problem

(out of three possible solutions). This exercise uses elements of pattern generalization and algorithm design, and falls under the "evaluate" category of Bloom's Taxonomy since students must justify why their chosen solution is most efficient. Lastly, "Terry" asks students to load cargo, which requires students to design a non-greedy algorithm to achieve the best answer, and falls under the "analyzing" level of Bloom's Taxonomy.

The five clues given to students upon completion of the five tasks include a set of ten faces and numbers, the number "3," a 6x6 grid with black and white shapes, and a modified Caesar cipher. One clue (the ten faces and numbers) is given to the students for "free" as part of the project packet. The faces can be sorted, and the number "3" can be used as an index to identify Sammy as the murderer. Alternatively, the 6x6 grid can be combined with the modified Caesar cipher, the parity CS Unplugged activity, and Sammy's data representation lesson to decode the grid and reveal Tammy as the murderer. The project was discovered to contain an error after it had been deployed, resulting in two characters (Sammy and Tammy) both being valid solutions for the murderer. The clues also have elements of CT concepts, including data representation, decomposition, and pattern recognition.

Dr. Tim Bell, the creator of the original CS Unplugged activities, provided feedback on the final project assessment and the experiment design. Dr. Bell provided his insight on the phrasing of the questions (i.e., rewording may better target the intended goal) and mapped each question to its CT skills. He also provided additional perspectives that had not yet been addressed, such as the fact that we deploy all the written materials in English only. His feedback is detailed in Chapter 6.

3.5 Bloom's Taxonomy

Similar to Thies and Vahrenhold's mapping from Bloom's Taxonomy to CS Unplugged activities [15], we mapped Bloom's levels of thinking to the six assessments (shown in Table 3.3). The mapping of Bloom's Taxonomy to the assessments was done in conjunction with two teachers from STEM School. Teachers individually evaluated the activities before discussing and justifying their choices. Independent analyses provided the opportunity to

find consensus among assignment placement. Notice that several of the assessments are located in the higher realms of Bloom's Taxonomy, which make them better poised to measure middle school learning than if they were unilaterally located in the bottom categories.

Table 3.3: The different Bloom's Taxonomy behaviors present in the CT assessments.

	Binary Numbers	Cryptology	Error Detection	Sorting & Searching	FSA	Final Project
Creating						X
Evaluating		X	X	X	X	X
Analyzing	X	X	X	X	X	X
Applying		X	X	X	X	X
Understanding	X		X	X	X	X
Remembering	X					X

3.6 Computational Thinking

Table 3.4 identifies the CT components tested in each portion of the final assessments. The table was constructed by having five members of our research group independently categorize the assessments before aggregating the results. All five members were familiar with the principles of computational thinking as well as CS Unplugged activities. Each of the five CT concepts is represented in one or more of the final assessments. Appendix C details the CT components of CS Unplugged activities used across all deployments and not just the final project.

Related research has already shown CS Unplugged activities to be engaging, but it is not known whether students are learning the desired computer science concepts. The purpose of deploying these activities and assessments in the classroom is to try and answer the two research questions proposed in this thesis. The associated worksheets and content assessments administered in-class will be used to determine what the students are taking away from each lesson and if students are understanding the main ideas of that activity's CS concept.

Table 3.4: The different CT components represented in each of the final assessments.

	Binary Numbers	Cryptol-ogy	Error Detection	Sorting & Searching	FSA	Final Project
Data Representation	X		X		X	X
Decomposition		X	X	X		
Pattern Recognition	X	X	X			X
Pattern Generalization & Abstraction		X		X	X	X
Algorithmic Thinking				X		X

CHAPTER 4
EXPERIMENTAL DESIGN

The CS Unplugged activities and their associated extensions were pilot tested and refined during the 2014 and 2015 school years. The final project underwent a small pilot test during the summer of 2015. The information gathered from the pilot test is summarized below. The deployment schedule for the fall semester is also outlined.

4.1 Final Project Pilot Test

A pilot test of the final project was performed in order to identify any major issues with the project content, to ensure the length of the project was appropriate, and to verify that the activities were engaging for students. To avoid inadvertent sharing of the mystery solution, the final project pilot test occurred in a different environment than the data collection deployment. An "Exploring Technology" summer session at Mines allowed the project to remain secluded from potential students attending STEM School. Exploring Technology students were first exposed to the same six CS Unplugged activities planned for the deployment at STEM School. After they had seen all six activities, half of the students were given the "Carnival" version of the final project, and half of the students were given the "Pet" version. In addition to collecting the final projects, several undergraduate student observers were present in the classroom taking observation notes while the students completed the final projects. After reviewing the observation notes and performing a high-level pass over the projects to see what was attempted and what was left blank, several alterations were made to the project before deploying it for data collection.

First, the final project contained too many components for most students to reasonably complete in 55 minutes. To combat this issue, the "Barry" character was removed from the packets before being deployed at STEM School. This change reduced the reading time substantially. Other smaller changes were made to individual worksheets, such as altering

the "Pet" version's optimization worksheet to use different numbers than the "Carnival" version (to prevent students from remembering the solution).

4.2 Deployment Schedule for Data Collection

Two groups were utilized to evaluate the computational thinking assessments: a retention group and a pre/post group. The retention group is marked as "Group 1" in Figure 4.1. Students in Group 1 were exposed to the six CS Unplugged activities first, took the "Pet" version of the final project as a posttest (signified by the dark outline in column G), returned to class taught by the regular instructor for six school days, then took the "Carnival" version as a retention test.

Group 2 completed the "Pet" version of the final project on the same day as Group 1. The difference is that Group 2 had not been exposed to any of the CS Unplugged activities yet (a pretest). After completing the first final project, Group 2 did the activities, and then took the "Carnival" version (a posttest) of the final project. In the deployment, both groups received the "Pet" version as their first final project, and the "Carnival" version as their second final project. The remaining columns and their associated lessons are described in Table 4.1.

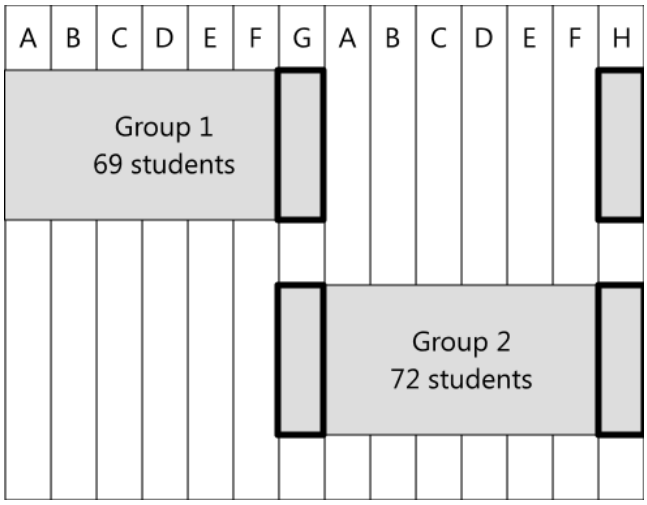


Figure 4.1: The fall 2015 deployment schedule. Each column is one school day, and each letter in a column represents a unique activity being deployed; see Table 4.1 for details.

Table 4.1: Order of Activity Deployment in Fall 2015

A	Binary Numbers
B	Caesar Ciphers and Frequency Analysis
C	Minimal Spanning Trees
D	Parity and Error Detection
E	Sorting and Searching
F	Finite State Automata
G	Pet Final Project
H	Carnival Final Project

CHAPTER 5

RESULTS

The results from the CS Unplugged deployment can be separated into three main categories: results from the in-class activity worksheets (Section 5.1), results from the final projects (Section 5.2), and results pertaining to the final project process (Section 5.3). The worksheets help gauge the effectiveness of the activities and support the argument that students learned the tools needed to complete the final projects. The final project scores help measure students' computational thinking skills and create a quantifiable means to compare student achievement. Results in the last section provide an indication of the level of student engagement and any issues encountered while the final project was being deployed.

5.1 Activity Worksheet Results

Six CS Unplugged activities were deployed to two groups of students. Each group consisted of three classes at STEM School and approximately 70 students. In each of the six activities, students completed worksheets in class. Unless otherwise noted, worksheets were completed individually by each student.

After the classroom deployment, a rubric for each of the worksheets was created. The rubrics provided guidelines on how to score questions on the worksheets as either "Proficient," "Partially Proficient," or "Unsatisfactory." Every worksheet collected from the classroom was individually scored by two researchers to ensure consensus. Disagreements on any score were resolved by having both researchers score the question together and editing the rubrics to better document any edge cases. The worksheets and related rubric for each activity are included as Appendices D through M.

The purpose of scoring and analyzing the worksheets is twofold. Analysis of student scores is used to (a) verify that the groups are comparable in knowledge attainment, and (b) determine whether students understood the concepts from the activities.

5.1.1 Proportion Test

A two-tailed proportion test was used to compare the results of Group 1 against Group 2. The proportion test was chosen because the student data is encoded into categories, the samples are independent, and proportions testing is appropriate for the student sample size. The proportion test requires binary data; thus, before running the proportion tests, the scored data was converted from three categories to two categories, which will be called *performing* (proficient and partially proficient) and *not performing* (unsatisfactory). A p-value of 0.05 was used to determine significance. All five of the activities share a common hypothesis for the proportions test: students from both groups should perform similarly on the worksheet(s) because each of the six classes were presented with the same information in the same manner. Absence of significant results is ideal in this case, as it would support that both groups received the same knowledge in preparation for the final projects.

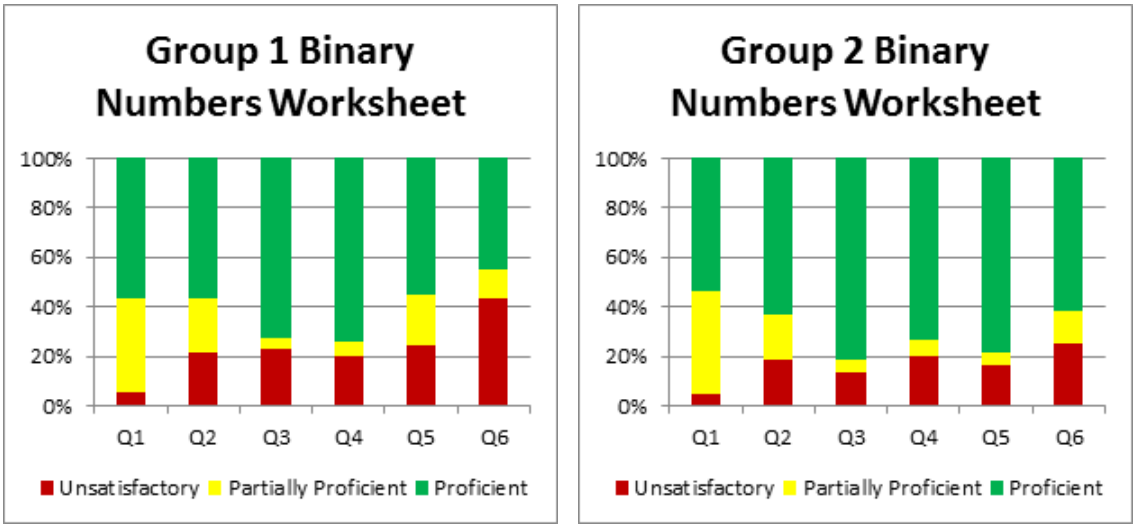
Appendix N contains abbreviated tables that describe what is being scored in each worksheet, and how each question relates to Bloom's Taxonomy and Computational Thinking. Questions that had statistically significant changes are marked with an asterisk in the right-hand column. Based on the proportion test, only one question on one of the worksheets had a significant result (Q6 of the "Binary Numbers" worksheet: How many bits are needed to represent 63?). With only one significant difference, the two groups appear to be comparable and had the same knowledge of CS concepts after seeing the activities.

5.1.2 Worksheet Analysis

In the following subsections, the results are presented as bar charts. Each chart contains the scores for one worksheet completed by both groups, unless otherwise noted. The bar charts display the percentages of students who scored proficient, partially proficient, and unsatisfactory for each item listed on the worksheet's associated rubric.

5.1.2.1 Day 1: Binary Numbers

Figure 5.1 shows the results of the binary number worksheet. Students demonstrated that they could recognize patterns of binary numbers (Question 1) as well as the ability to convert between binary and decimal number systems (Questions 2 and 3). Questions 2 and 3 had over 70% of students scoring either proficient or partially proficient. Students also understood the range of numbers that could be represented with five bits (Question 4), which was a large focus of the classroom presentation. Questions 5 and 6, which dealt with the more general case of mapping the number of bits to a numeric range, had the highest ratio of unsatisfactory responses. This result is not surprising since these two questions fall on the upper scales of Bloom's Taxonomy and of computational thinking ability. Students in Group 2 did significantly better than the students in Group 1 on Question 6; however, Question 6 was the only question in this worksheet where both groups had less than 80% score in the proficient or partially proficient categories.



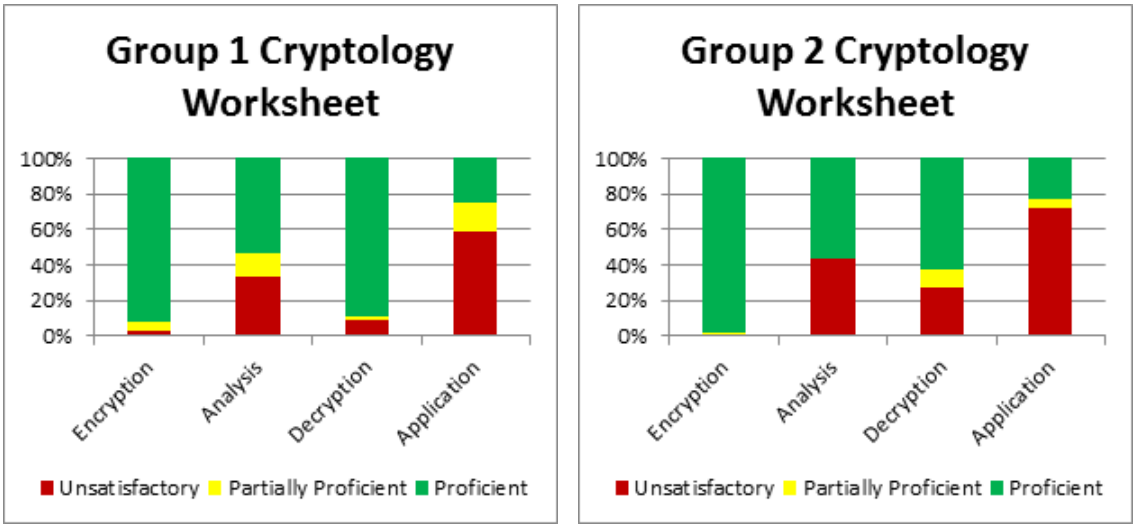
(a) Group 1; 60 submissions

(b) Group 2; 69 submissions

Figure 5.1: Results for the Binary Numbers "Check Your Understanding" worksheet.

5.1.2.2 Day 2: Caesar Ciphers and Frequency Analysis (Cryptology)

Students completed two worksheets as part of the activity on Caesar Ciphers and Frequency Analysis. Figure 5.2 shows that the majority of students were comfortable using ciphers to encrypt and decrypt messages, and could also determine the number of possible keys in a Caesar cipher since more than 75% from both groups were able to attain partially proficient or proficient status on those problems. Figure 5.2 shows 59% of students in Group 1 and 71% of students in Group 2 scored in the unsatisfactory range on the last problem. This problem was time consuming because it required students to decrypt a message based solely on frequency analysis. This task may not be a realistic assessment of students' learning because they did not have enough time to complete the problem.



(a) Group 1; 65 submissions (b) Group 2; 65 submissions

Figure 5.2: Results for the two worksheets used in the Caesar Cipher & Frequency Analysis activity.

5.1.2.3 Day 3: Minimal Spanning Trees

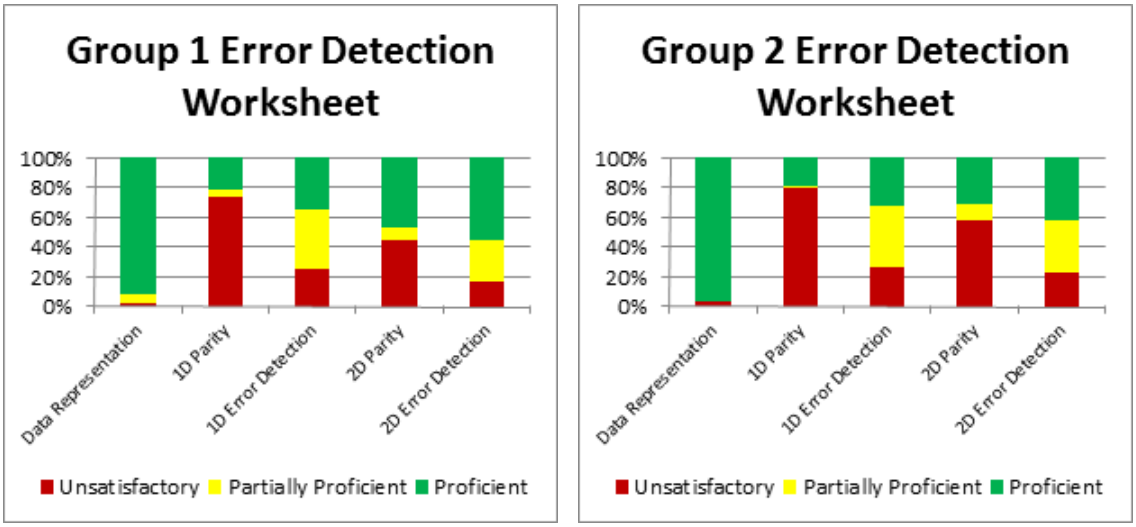
The two worksheets for the "Minimal Spanning Tree" activity were slightly different in the fact that one of the worksheets was laminated (a classroom set reused each period) and the other worksheet did not effectively capture the students' thought process while they

solved the problem. The design of the "Minimal Spanning Tree" worksheets made them unsuitable for analysis.

5.1.2.4 Day 4: Parity and Error Detection

Figure 5.3 shows the results for the five different problem areas on the error detection worksheets. Students overwhelmingly scored well on the data representation problem, clearly showing comfort in representing letters as numbers. The majority of students did not attempt the problem related to 1D parity. Only 19 students out of 64 attempted the problem in Group 1, and 17 out of 70 attempted it in Group 2. Of the students who did attempt the 1D parity problem, the majority appeared to grasp the concept of a parity bit. The instructions on the worksheet for this question may not have been clear to most students, as only 36 students across both groups actually attempted the problem.

Students scored much better on the error detection portions of the worksheet. This result suggests that students can apply an error detection algorithm to data that has the parity bits added, but struggle to initially compute what value the parity bit should be.



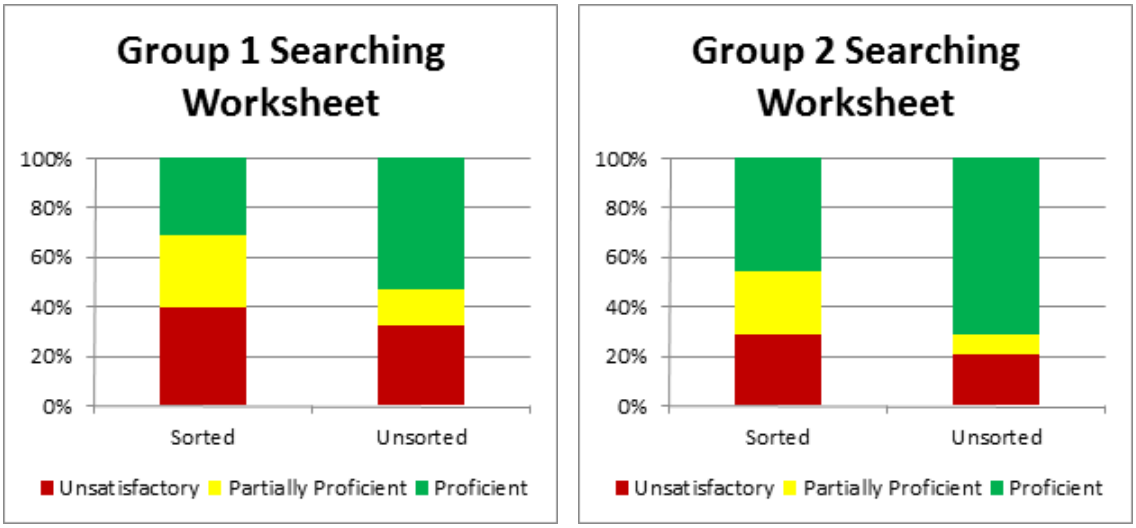
(a) Group 1; 64 submissions (b) Group 2; 70 submissions

Figure 5.3: Results for the two worksheets used in the Parity and Error Detection activity.

5.1.2.5 Day 5: Searching and Sorting

Searching and sorting were both covered in one class period, but the results here are displayed in separate charts and tables. All of the activities in this lesson were done by pairs of students, so one submission constitutes two students.

As seen in Figure 5.4, a higher percentage of students were in the unsatisfactory range compared to other activities. A potential issue with this activity is that the students did not have any materials at their desks to individually practice the searching and sorting algorithms before attempting the worksheet. Regardless, over 60% of students were able to reach proficient or partially proficient categories for both groups. For the unsorted data, acceptable answers included random or linear searching, whereas for the sorted data, the correct answer was binary or linear searching. This lenient scoring scheme may help to explain the gap in performance between the two columns. The other half of the searching and

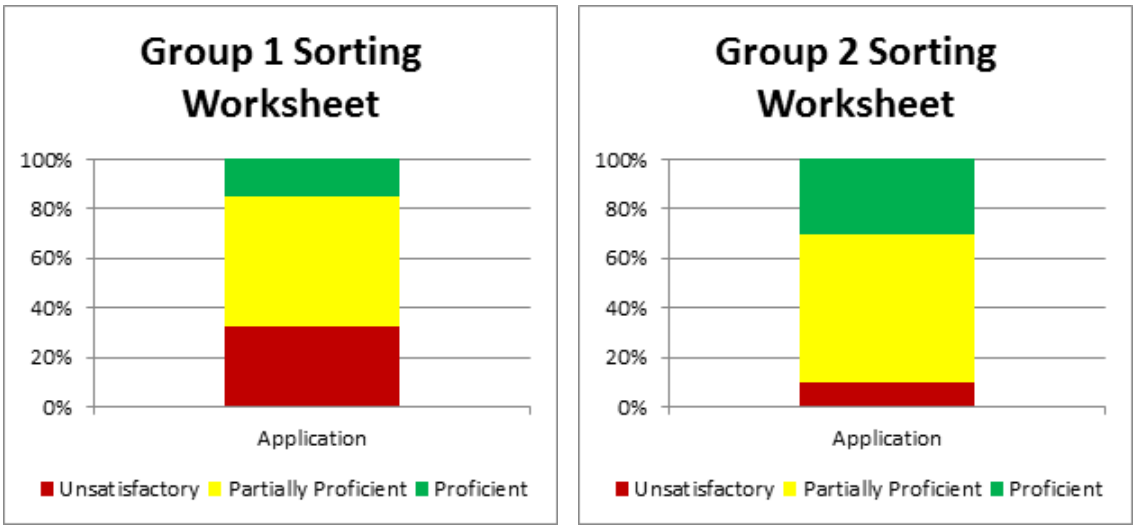


(a) Group 1; 24 submissions (b) Group 2; 34 submissions

Figure 5.4: Results for the Searching worksheets used as part of the Sorting and Searching activity.

sorting activity involved a classroom demonstration of quicksort and selection sort. Students had a harder time applying the sorting algorithms (which must be used to attain proficiency) compared to using a brute force approach until the six colors are sorted. Less than 35% of

students were able to successfully re-apply a sorting algorithm from the demonstration to this worksheet (see Figure 5.5).



(a) Group 1; 30 submissions

(b) Group 2; 34 submissions

Figure 5.5: Results for the Sorting worksheet used as part of the Sorting and Searching activity.

5.1.2.6 Day 6: Finite State Automata (FSA)

Students completed two worksheets as part of the FSA activity. Unfortunately, a communication error resulted in the worksheets for Group 1 not being collected. For this reason, we only present the results for Group 2 in this section. Figure 5.6 suggests that students were comfortable selecting appropriate states for use in an FSA diagram, and were somewhat comfortable completing transitions on an already connected FSA graph. The majority of students did not attempt the "Transitions" problem of the FSA worksheet because they focused their time on solving the other two problems on the worksheet. While students were comfortable completing isolated parts of an FSA problem, the high number of unsatisfactory responses on the "Finite State Construction" challenge indicated that combining the state selection and transition completion steps is an area where most students struggled. The "Transitions" column in Figure 5.6 represents the 10 students who attempted the problem,

whereas the other two columns represent the full set of 66 responses.

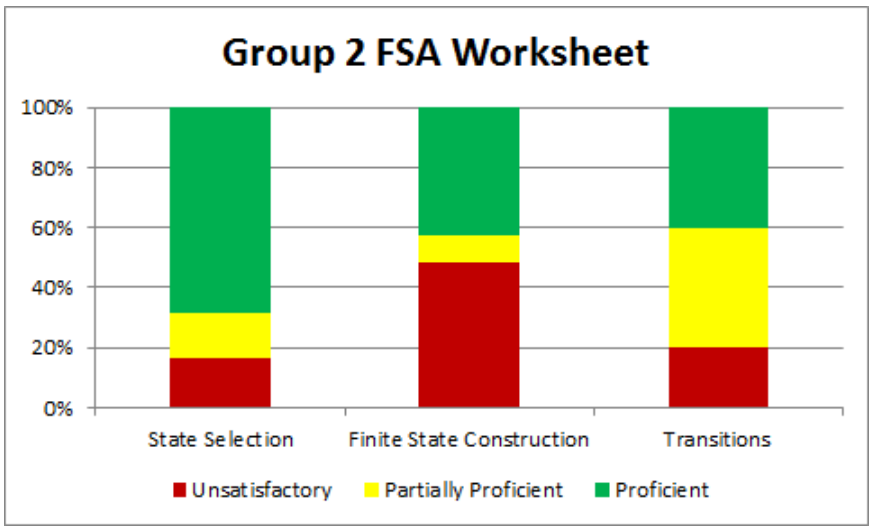


Figure 5.6: Group 2's results for the Finite State Automata worksheets. Note the first two columns represent the scores of 66 student attempts. The third column consists of 10 student attempts (the remaining 56 students did not attempt this problem).

5.2 Final Project Comparisons

The following subsections present the results from the final projects. The first section shows two final project comparisons and their associated statistics. The second section shows three final project comparisons that are not suitable for statistical analysis, but which highlight interesting comparisons. The third section is the most directly relevant section for the research questions, and relies on two types of statistics to interpret the results.

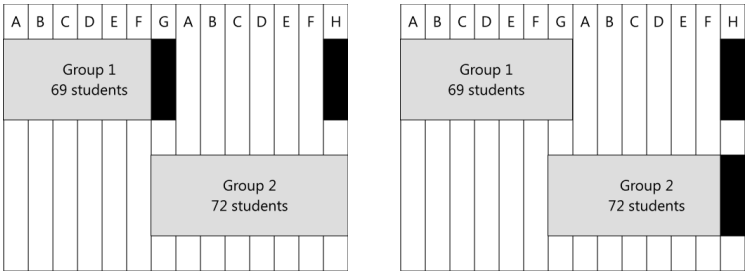
The final projects completed by each student were paired before being anonymized. This is different from the analysis of the worksheet results where each worksheet was only completed once. Perfect matching is unlikely due to the chance of student absences and the chance of a student forgetting to write their name on the project packet. In Group 1, 60 out of the 69 students were able to be paired from their first final project to their second final project. In Group 2, 55 out of the 72 students were able to be paired. In total, the final project data set contains 115 data points.

The final projects and associated rubrics used to score the projects are attached as Appendices O through T. The process of scoring each final project was the same as the process for scoring the activity worksheets described in Section 5.1. The following sections present bar charts that show the percentage of students who attained full proficiency and mastery of the question domains, and pie charts that provide a breakdown between unsatisfactory, partially proficient, and proficient for any statistically significant results.

5.2.1 Statistically Testable Comparison Results

A χ^2 test was used with three of the final project comparisons; two of the comparisons are discussed in this section and the third comparison is discussed in Section 5.2.3. The χ^2 test was used because the final project scores were categorical (proficient, partially proficient, or unsatisfactory). The χ^2 test takes into account all three scoring categories across both groups being compared, and assesses the goodness of fit between those observed values and calculated expected values. A significant result tells us that student performance changed beyond what can reasonably be attributed to chance.

Figure 5.7 illustrates the deployment schedule and highlights the comparison being made. Note that the deployment schedule is the same as shown in Figure 4.1 - the only difference is the comparison. In Figure 5.7(a), the comparison is between the posttest and retention tests of students in Group 1. Figure 5.7(b) shows the comparison between the retention test of Group 1 and the posttest of Group 2. Figure 5.7(a) is classified as an intragroup comparison because both finals in the comparison were taken by the same group of students. Figure 5.7(b) is classified as an intergroup comparison because the comparison involves both groups of students. Table 5.1 shows the χ^2 results for two of the three comparisons, and Table 5.2 shows the proportion test results for the two intragroup comparisons. Results will be discussed in the following sections. For problems with a significant result, associated pie charts are shown that provide a complete breakdown between unsatisfactory, partially proficient, and proficient scoring categories.



(a) Group 1 (b) Retention Test - Posttest

Figure 5.7: Final Project comparisons used with the χ^2 test

Table 5.1: Final Project Comparison χ^2 Test Results. Significant results marked in bold.

	Posttest and Retention Test p-value	Retention Test and Posttest p-value
Clue #1	0.44861	0.39754
Binary Numbers	0.401729	0.574965
Searching	0.093315	0.087272
MST	0.853057	0.259799
FSA Data Rep	0.246791	**0.040175**
FSA Application	0.377192	0.102505
Optimization	**6.91E-8**	0.309138

5.2.1.1 Group 1 Posttest - Group 1 Retention Test Results

This comparison answers the question "Do students retain information from the CS Unplugged activities after a delay?" When comparing two tests taken after seeing the activities, they should ideally have equivalent results. Taking into consideration that students went back to their regular class work for six days between the final projects, a drop in proficiency is understandable. Figure 5.8 shows the change in proficiency in Group 1 between the "Pet" final (posttest) and the "Carnival" final (retention test). As expected, the number of students attaining proficient scores dropped slightly between the two final projects, with the exception of the optimization problem, which saw tremendously higher achievement in the retention test. The optimization problem is the only significant difference in the comparison

Table 5.2: Final Project Comparison Proportion Test Results. Significant results marked in bold.

	Posttest and Retention Test p-value
Clue #1	N/A
Binary Numbers	1.430458
Searching	1.701544
MST	1.23102
FSA Data Rep	0.569541
FSA Application	0.591208
Optimization	0.569541

according to the χ^2 test. The full breakout of student scores among the three grading cate-

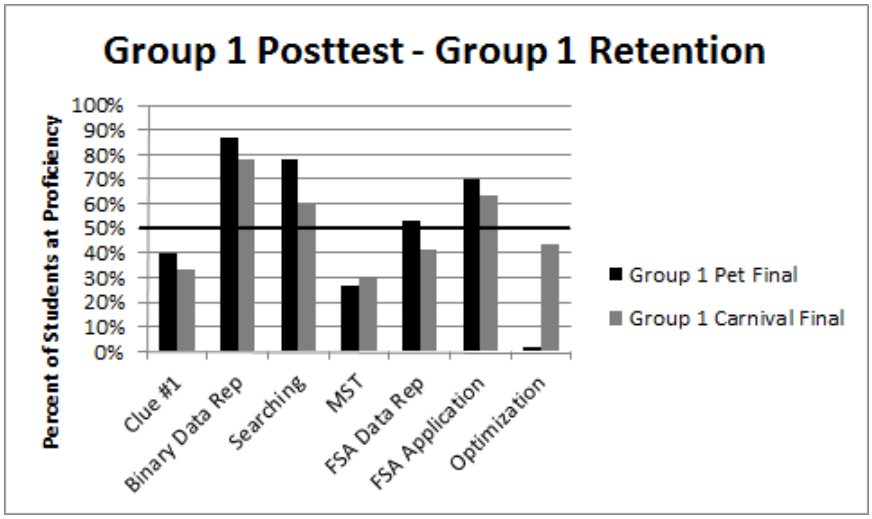
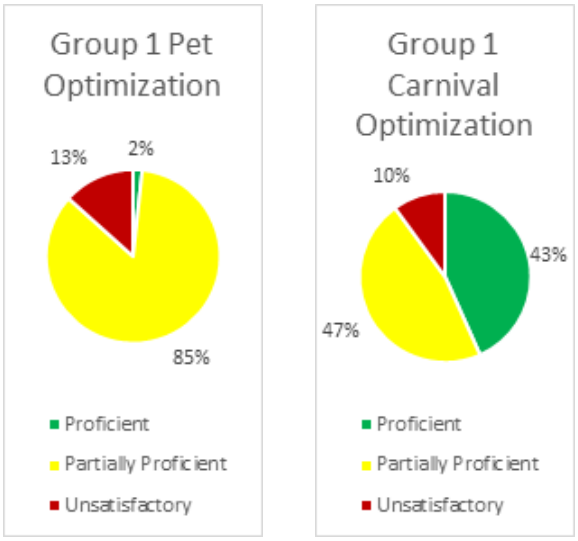


Figure 5.8: Chart of Group 1 proficient scores in the posttest and retention test.

gories for the optimization problem is shown in Figure 5.9. This result will be discussed in Chapter 6.

5.2.1.2 Group 1 Retention Test - Group 2 Posttest Results

This comparison answers the question "Do students who have had a delay between completing the activities score as well as students who have just seen the activities?" This project



(a) Pretest (b) Posttest

Figure 5.9: Breakout of Group 1 student performance on statistically significant project problems.

comparison measures the retention test from Group 1 to the posttest from Group 2. This comparison is appropriate for the χ^2 test because both groups are taking the same version of the final project and both groups of students have seen the activities before completing the project packet. When comparing two projects taken after seeing the activities, proficiency should be equivalent. Taking into consideration that one of the groups had gone back to regular class work for six school days, it would also be understandable if the retention group had lower proficiency on the final project than the posttest group.

Figure 5.10 shows student proficiency between the retention test and the posttest. This project comparison is interesting because the retention test scores are higher than the posttest in six of the seven project categories. The posttest group did have more students reach proficiency in the minimal spanning tree problem.

This comparison did have one statistically significant result for the FSA Data Representation problem. The retention group did significantly better than the students in the posttest (category breakouts can be seen in Figure 5.11).

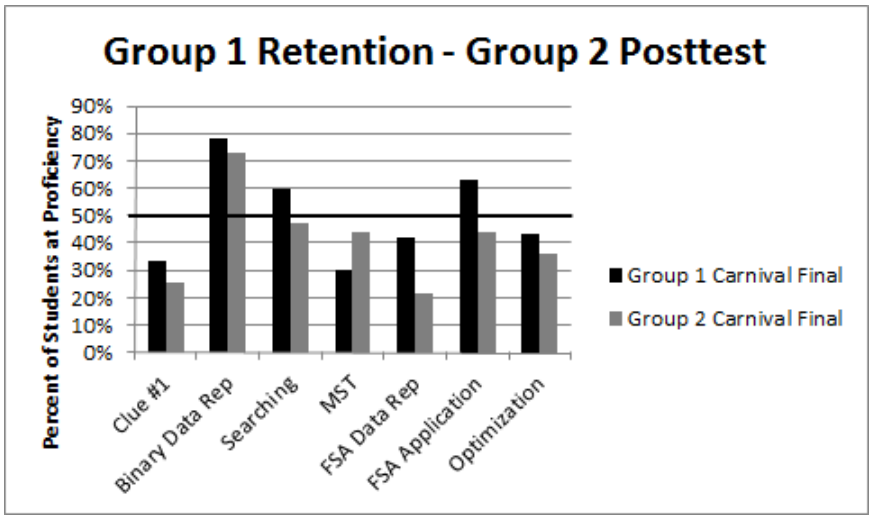
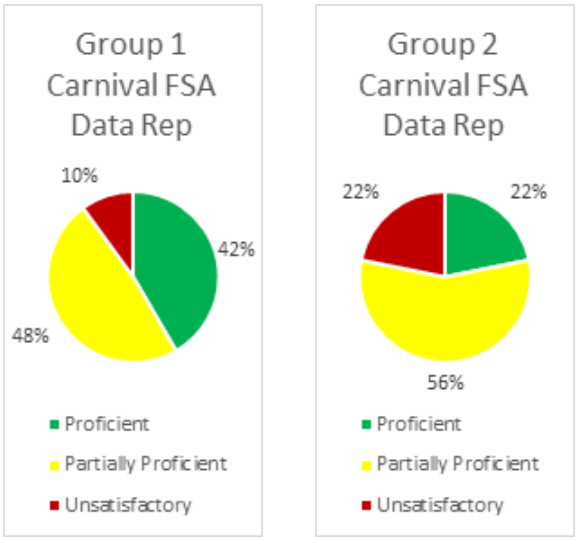


Figure 5.10: Chart of students who scored proficient in Group 1's retention test and Group 2's posttest

5.2.2 Intergroup Comparison Results

The following three final project comparisons are similar in design to those in the previous section, but without the χ^2 test. These comparisons offer additional perspectives that reinforce the observed student learning from the CS Unplugged activities. The comparisons in this section contain too many independent variables (different groups of students, different versions of the final project, or assessments deployed at different stages of the learning process) for meaningful statistics to be extrapolated. The same categorical scoring system used in the previous section was also used in this section.

Figure 5.12 illustrates the three comparisons whose results are presented in this section. Figure 5.12(a) shows the comparison between the posttest from Group 1 and the pretest from Group 2 (the same final project). Figure 5.12(b) shows the comparison between the retention test from Group 1 and the pretest from Group 2 (two different final projects). Figure 5.12(c) shows the comparison between the posttest from Group 1 and the posttest from Group 2 (two different final projects).



(a) Pretest (b) Posttest

Figure 5.11: Breakout of intergroup student performance on a statistically significant project problem.

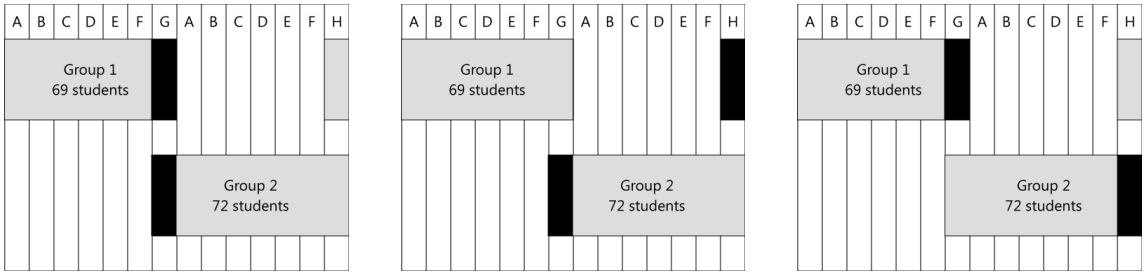
5.2.2.1 Group 1 Posttest - Group 2 Pretest Results

This comparison answers the question "Do students who have seen the activities do better on the final project than students who have not?" We expected the posttest group to score higher than the students in the pretest group since the students had already been exposed to the CS Unplugged activities for the posttest. Although no statistical analysis can be performed, this hypothesis appears to hold true for all components of the final project except the optimization problem. This result suggests that the CS Unplugged activities are meaningful in teaching their intended CS concepts.

The majority of students taking the posttest were able to reach proficiency in four of the seven components (see Figure 5.13). In contrast, the pretest group attained majority proficiency in only one problem area (binary data representation).

5.2.2.2 Group 1 Retention Test - Group 2 Pretest Results

This comparison answers the question "Do students who have seen the activities perform better, even after a delay (i.e., retention), than students who have never seen the activities?"



(a) Posttest - Pretest (b) Retention Test - Pretest (c) Posttest - Posttest

Figure 5.12: Intergroup Comparisons. Black bars in each subfigure mark two of the dates when final projects were deployed and the semantic relation between the two groups.

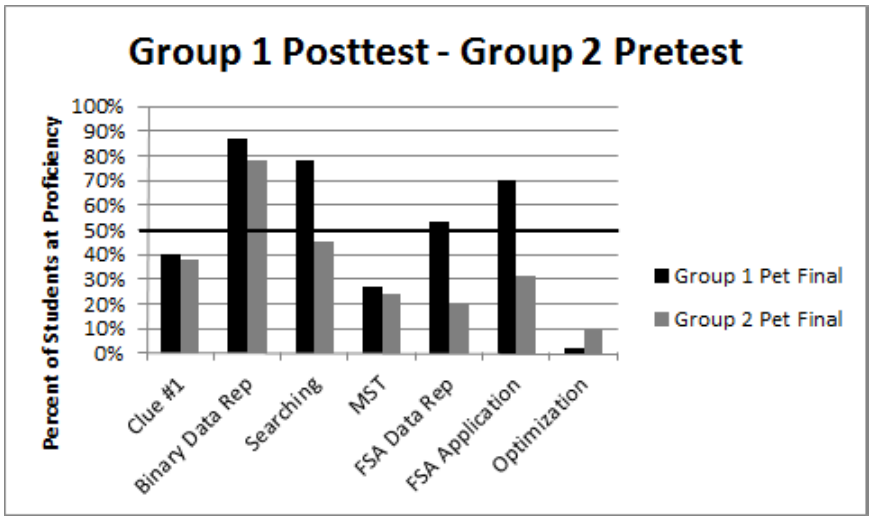


Figure 5.13: Chart of students who scored proficient in Group 1's posttest and Group 2's pretest

Figure 5.14 shows that in every category except Clue #1, we do see that the retention scores were higher than the pretest. This is an exciting result because it shows that students are retaining the information after they've learned it, which reinforces the entire idea behind teaching computational thinking in middle schools.

5.2.2.3 Group 1 Posttest - Group 2 Posttest Results

This comparison answers the question "Do two independent groups who have just seen the activities score similarly on the final project?" We expected student proficiency to be

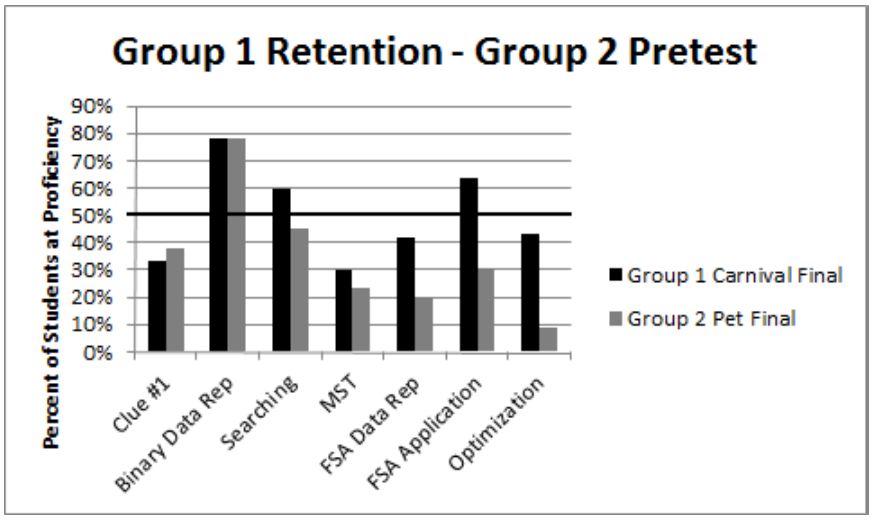


Figure 5.14: Chart of students who scored proficient in Group 1's retention test and Group 2's pretest

equivalent among group posttests because both groups were at the same stage in their learning progressions. Figure 5.15 shows that Group 1 had a greater percentage of students reach proficiency than Group 2 in five of the seven components. In addition, Group 1 was able to have the majority of students reach proficiency in four of the seven components, whereas Group 2 only achieved majority proficiency in the binary data representation problem. This result is somewhat surprising, since the worksheet results show the two groups had roughly equivalent knowledge.

5.2.3 Pretest to Posttest Comparison and Statistics

Figure 5.16 shows the comparison between the pretest and posttest of students in Group 2. This comparison answers the question "Are students learning information from the CS Unplugged activities?" This comparison is most directly related to the research question "Do CS Unplugged activities encourage computational thinking?" It relies on both the χ^2 statistic and the proportion test for significant results. Student proficiency would be expected to increase from a pretest to a posttest since the students would have learned the necessary content between completing the two final projects. Figure 5.17 shows that the number of

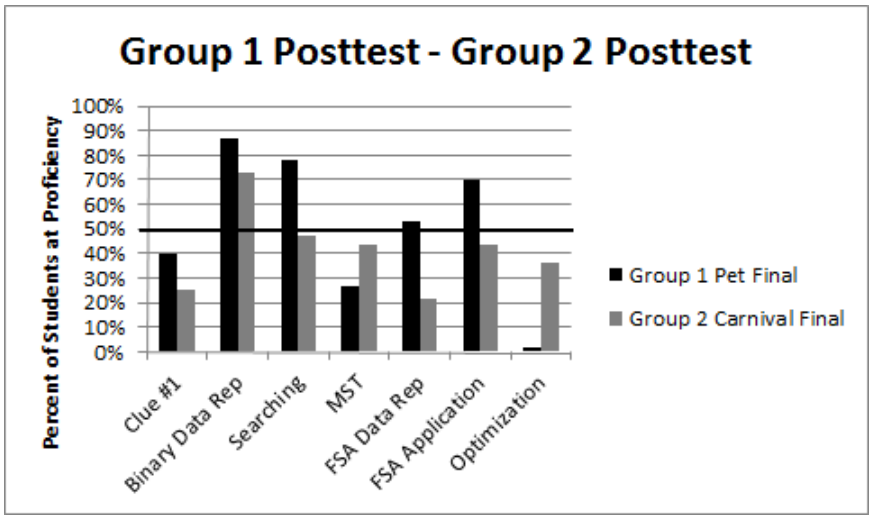


Figure 5.15: Chart of students who scored proficient in Group 1's posttest test and Group 2's posttest

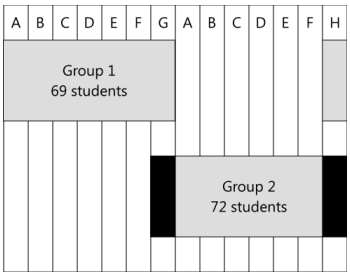


Figure 5.16: Pretest-Posttest Comparison for Group 2.

students who scored proficient did increase in five of the seven project categories, and three of these were significant. "Clue #1" and "Binary Data Representation" were the only two problems that showed a decrease, although neither component was a statistically significant drop. That is, the binary data representation category only decreased from 78% to 73%, and Clue #1 only decreased from 38% to 25%. The binary data representation problem, despite having a drop in student proficiency, is still the only component of the final project where more than half of the students in Group 2 were able to achieve proficiency. This result is surprising given that students who had never seen the binary numbers activity scored so highly. In reviewing the final projects, the "Pet" final provides more specific instructions so students would not get stuck, and it may have skewed the results.

Table 5.3: Pretest-Posttest Comparison Statistical Test Results. Significant results marked in bold.

	χ^2 Test p-value	Proportion Test p-value
Clue #1	0.355007	1
Binary Numbers	0.670072	1
Searching	0.91318	1.175337
MST	0.072341	0.207252
FSA Data Rep	0.104719	**0.039089**
FSA Application	**0.046312**	**0.013290**
Optimization	**0.002870**	0.189084

Students in Group 2 demonstrated a significant increase in "FSA Data Representation," "FSA Application" and the "Optimization" problems of the final project. Despite the measured increase, less than half of the students in Group 2 achieved proficiency in these three problems. The "FSA Data Representation" change is not apparent in Figure 5.17 because the chart only shows full proficiency. Figure 5.18 displays pie charts with all three scoring categories where the significant change is more obvious. The significant results from this comparison suggest that students are *learning* the intended information from the CS Unplugged activities, and that the final projects are capturing the growth in performance.

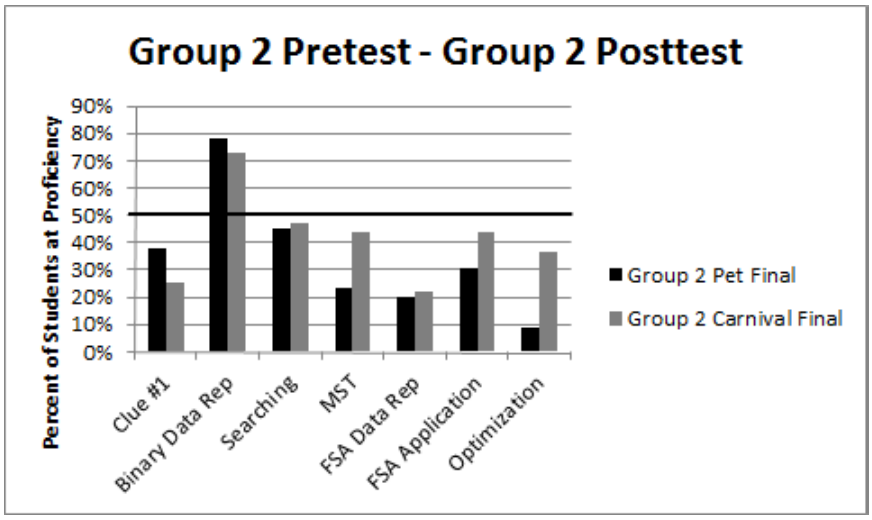


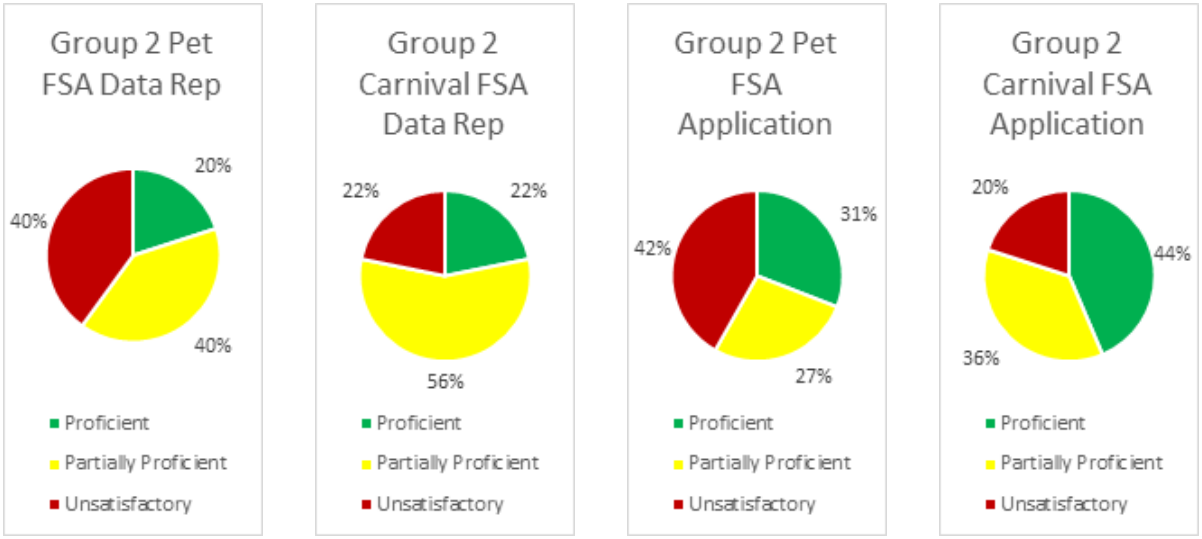
Figure 5.17: Chart of Group 2 proficient scores in the pretest and posttest.

5.3 Final Project Process Results

Five to six Mines student observers were in the classrooms during the final project deployments. The observers helped facilitate the final project, and recorded notes on the behavior of the classroom and the questions asked by students. These observations are used to determine student engagement and identify any issues in the project content that was not evident during the scoring process.

For the "Pet" final, all of the observers noted that the students were working "diligently" and "quietly," with little to no chaos or interruptions. Students were engaged in the final project and asked thoughtful questions relevant to solving the problems. For example, multiple students taking the pretest in Group 2 asked if they could use a calculator to solve the minimal spanning tree problem, which is understandable since they did not yet know Kruskal's algorithm. Other students asked if the problem illustrations were drawn to scale (on both the optimization problem and the minimal spanning tree problem). Although everything was labeled, none of the problems explicitly stated they were not drawn to scale, which could hinder a student's interpretation of the problem. There were also questions about the FSA problem during the pretest, as students weren't quite sure how to begin solving it.

In the "Carnival" final, students again were mostly working quietly and in an organized fashion, as noted by observers. There were several students who did not remember or could not make the connection between the binary number representation problem in the project and the binary numbers activity they had seen prior to the project. One student also noted minor wording differences on one of the clues – in the "Pet" version of the final project, the Caesar cipher mentioned it was an "encoding" cipher, which was useful for solving the problem. The "Carnival" final project did not mention this fact, which may have hindered the student had the student not remembered the additional information from the previous final project.



(a) Pretest (b) Posttest (c) Pretest (d) Posttest

(e) Pretest (f) Posttest

Figure 5.18: Breakout of Group 2 student performance on statistically significant project problems.

CHAPTER 6

DISCUSSION

Chapter 5 explored results from the activity worksheets and from final project comparisons, and we saw these comparisons produced multiple positive statistically significant results. In addition to the results that supported the stated hypotheses, some results appeared abnormal or unexpected. In this section, we dedicate time to discussing the results that were unexpected and how the assessments or activities might be modified to address these outcomes.

6.1 Sorting and Searching Activity Deployment

The "Sorting and Searching" activity was deployed in one class period and covers four different algorithms: linear search, binary search, quicksort, and selection sort. To cover all of this material in 55 minutes, class demonstrations were used as opposed to individual problems for students to practice on. The overwhelming amount of information in this lesson could easily cause students to be confused. In fact, sorting and searching are usually taught separately at the university level.

Student comprehension and scores in these two areas would likely improve if each subject were expanded to fill its own class period. Several students wrote answers on the final project that alluded to their confusion between the two different problems. "Binary sort" was given as an answer to Clue #1 several times (which caused a bump down to partially proficient), and "quicksort" was written on a project to explain their solution to the searching problem. Separating these two activities would allow more time for students to practice the algorithms and differentiate that sorting and searching are two independent problems with different constraints.

6.2 Cryptology Activity Deployment

This activity covered Caesar ciphers well and introduced the concept of substitution ciphers and frequency analysis. One lesson learned from the analysis of the worksheets was that students did not have enough time for meaningful progress to be made on the last problem of the worksheet (using frequency analysis to decrypt a ciphertext message). The ciphertext message could be altered to be shorter or easier to perform frequency analysis on (e.g., more word repetition).

6.3 Parity & Error Detection Activity Deployment

The worksheets for this activity suggested that students struggled to calculate the parity bit of a message themselves. Error detection is quite an advanced topic to begin with, so it may also be worthwhile pilot testing a version of the activity that only deals with 1D parity and error detection, and leaves the 2D versions of the problems as a challenge for students who finish early. One drawback would be that the engaging magic trick relies on 2D parity. Another option would be to place more emphasis and practice on computing the value of a parity bit so students fully understand how the larger idea of error detection works.

6.4 Binary Data Representation of Final Project

The binary data representation problem of the final projects had very high scores from all students across both final projects. Those results were unexpected and suggest a deeper methodological error in the final project. The problem in the "Carnival" version does not mention a binary system by name, but does give the hint that a "base-2" number system is being used. Additionally, the first binary encodings of the letters were given to the student, further providing pattern information even if they did not remember the material.

The "Pet" version of the problem was similar to the "Carnival" version, but more instruction was provided and the representation of the binary numbers was modified. These changes were done to allow the student to complete the problem without getting completely stuck, as we believed the alternative result would have been frustrating for the pretest group. In ret-

rospect and additional analysis of the problem after the deployment, the hints provided are likely responsible for skewing the data and allowing many more students to score proficient than if they had been given no hints at all.

6.5 Optimization Problem of Final Project

The optimization problem of the final project returned significant results for both groups from both the χ^2 test and the proportion test. These results look extremely out of place, as the optimization problem improved by great margins when other problems showed a decrease in student proficiency. The two optimization problems in both versions of the final project were almost identical, but with different numbers being used. We have several theories as to why the discrepancy in scores occurred:

- The "Pet" version of the problem stated that the goal was to place as many *animals* into each container as possible, when the actual goal was to place as much *weight* into each container as possible, up to the constraint. Several students noticed this confusing instruction and asked for clarification, bringing it to our attention.

- The "Carnival" version's optimal solution included one container at the maximum weight. We believe that in the process of maximizing the answer, the problem becomes more trivial when you are able to reach the maximum weight in one container; the challenge is reduced to finding the maximum weight you can place in the second container. The "Pet" version's optimal solution had both containers filled with weights less than the maximum allowed value. Therefore, the problem is not analogous to the "Carnival" version where students were able to reduce the problem to an easier challenge.

One way to correct this problem is to modify the "Pet" version of the final project such that it reduces down to the same problem the "Carnival" students face, or to modify the "Carnival" version of the final project such that both containers have less than the maximum

value in the optimal solution. If the problems are analogous, the results will likely be more meaningful.

6.6 Differences Between Student Groups

Group 1 scored surprisingly well on the final projects despite our initial hypotheses that the two groups would be equivalent. Here we postulate potential contributing factors to Group 1's scores that may have impacted their focus on the final project.

- Group 1 consisted of morning classes. Students were less social in their first class of the day versus their third or fourth.

- Group 2 consisted of morning and afternoon classes. Group 2 included one period that was 80 minutes long to accommodate a 25 minute lunch in the middle of class. There was no time allotted for a passing period between lunch and class, so the passing period was cut from time spent teaching the lesson plan (e.g., students were in class for 25 minutes, left for lunch for 25 minutes, then shuffled in for about 20 more minutes of class). The lunch break proved to be a huge interruption, as students quickly lost interest in the minutes leading up to lunch and stopped paying attention. This fact was evident in the numerous "It's lunch time, can we go now?" questions asked by students.

- Group 2 included one class that was completely full. All chairs were taken, and when the deployment began in early September, the students were on their third seating chart (as the first two were unsuccessful in controlling the chatter among students). Time was lost on this class waiting for the students to quiet down before moving on to subsequent activities.

- Being a STEM school, students were allowed to use their cell phones to listen to music in class. This rule was suspended for the duration of the CS Unplugged deployment, but one class in Group 2 had a difficult time parting with their devices. This resulted

in a ban of phones in the classroom, and any phones seen outside of backpacks were confiscated until the end of class. Confiscating a phone usually meant the student would stop participating for the rest of the day in retaliation (even though that meant they would not receive the participation points granted for the activity).

- Students in one Group 2 class that were especially loud and disruptive had emails and phone calls sent to their parents in the middle of the deployment. Classroom behavior and organization appeared to improve for the latter half of the activities, resulting in more students focusing on the worksheets and activities at hand. Several students apologized for their behavior following the intervention.

There were many factors not related to the experiment design that occurred in Group 2. The complete ban of cell phones may have incentivized some students to stop paying attention all together in class, which was likely a factor in the resulting phone calls home. While the group ended on a strong note, the turbulent deployment could have adversely affected their performance on the second final project ("Carnival" version).

6.7 Feedback from Tim Bell

Input on the final projects was solicited from Dr. Tim Bell, the creator of the original CS Unplugged activities. Dr. Bell contributed to our project in two distinct ways: he provided general feedback on the overall assessment and experiment design, and he mapped each question of the activity worksheets and final project to its associated CT skill. We received feedback from Dr. Bell after the deployment had finished in fall of 2015, so the projects administered to students did not take into account his suggestions.

Dr. Bell noted that some students may have seen or heard of some of the problems in the final project before. As an example, perhaps a student knows that for 1,000 items, you need to probe it 10 times to search the entire list. Using a "less common" value, such as 4,000, would mitigate the problem. The same issue arose with the cryptology activity: students may have seen a Caesar cipher before the CS Unplugged activities. Using a slightly

modified cipher (such as the one used as a clue in the final projects) may differentiate the problem sufficiently from what a student has seen before. Lastly, he commented on the wording of some of the questions and how they might distract students in unintended ways. One example would be the minimal spanning tree problem in the final projects - a carnival building a railroad is a highly uncommon idea, and might confuse students on what the problem is actually asking.

Dr. Bell also evaluated all of the assessments we were using as part of our deployment. He is an expert in the field and an accomplished researcher in computational thinking, so his classification of the assessment problems was valuable in understanding what skills we meant to test and what skills we were testing. Appendix N contains tables that present each question and its associated CT skill.

CHAPTER 7

CONCLUSIONS

The goal of this research project was to answer two questions. The following two sections discuss the conclusions that can be drawn from the results and methodology of the work presented as part of this thesis.

7.1 Question 1: Can we develop an effective instrument to determine what CT principles students are acquiring from the kinesthetic CS Unplugged activities?

The final project consisted of seven problem categories, two of which ended with methodological errors (the binary data representation and the optimization problems). This left five problems that were useful in assessing content from the CS Unplugged activities, but the scores collected using these problems were generally lower than what would be ideal. The final project needs to be revised to better target the 7th grade age group and allow for more students to achieve full proficiency on the problems.

The design of the final project is still useful. The project itself drew on various other research projects, and the scaffolded design is worth revisiting in the future. Allowing the activities to be independently completed of one another prevented the two problems with methodological errors from contaminating the results of the other five problems. Removing the dependency between the different categories also allowed for better isolation of the computational thinking skills being assessed.

7.2 Question 2: Do CS Unplugged activities encourage computational thinking?

The worksheets and extensions for the activities support the conclusion that CS Unplugged activities do encourage computational thinking. Each question on the worksheets was mapped to a computational thinking skill (Appendix N). Analysis of the scored work-

sheets showed that the students were understanding the concepts being targeted by many of the questions, and thus the underlying CT principles of that activity.

Students demonstrated mastery of at least one problem domain from each CS Unplugged activity they completed. After learning about binary numbers, students were able to convert between a base-2 number system and a base-10 number system. After the cryptology lesson, students understood how to take a message and encrypt it to make a jumbled piece of text, and decrypt encoded messages to see what someone else was trying to say. Real world connections in several of the activities also work to ensure students can relate to the importance of the skills they are learning and understand examples of the skills in their everyday lives. Students, even in 7th grade, are familiar with passwords, but most did not understand what role a password plays in protecting or encrypting data until learning about ciphers.

Other notable areas of student progress included the minimal spanning tree activity - students in the pretest group asked to use a calculator to solve the problem, whereas no students in the posttest group asked for a calculator because they knew Kruskal's algorithm. Students also were able to do (at the most basic level) analysis of algorithmic complexity by counting the number of comparisons needed to sort a list using quicksort versus selection sort, and similarly for binary search versus linear search.

REFERENCES CITED

[1] J. M. Wing. Computational thinking. *Communications of the ACM*, 49(3):33–35, 2006.

[2] Computer Science Teachers Association. Operational definition of computational thinking, 2011. URL http://csta.acm.org. Accessed 9 April 2015.

[3] National Science Foundation. NCSES science and engineering degrees: 1966-2010, June 2010. URL http://nsf.gov. Accessed 9 April 2015.

[4] K. Brennan and M. Resnick. New frameworks for studying and assessing the development of computational thinking. In *American Educational Research Association Meeting*, Vancouver, 2012.

[5] I. Lee and K Apone. Integrating computational thinking across the K-8 curriculum. *ACM Inroads*, 5(4):64–71, December 2014.

[6] S. Grover, S. Cooper, and R. Pea. Assessing computational learning in K-12. In *Conference on Innovation & Technology in Computer Science Education*, Uppsala, 2014.

[7] R. Thies and J. Vahrenhold. On plugging "unplugged" into CS classes. In *Special Interest Group on Computer Science Education*, Denver, 2013.

[8] L. Lambert and H. Guiffre. Computer science outreach in an elementary school. *Journal of Computing Sciences in Colleges*, 24(3):118–124, 2009.

[9] L. Seiter and B. Foreman. Modeling the learning progressions of computational thinking of primary grade students. In *International Conference on Computing Education Research*, San Diego, 2013.

[10] M. Friend and R. Cutler. Efficient egg drop contests: How middle school girls think about algorithmic efficiency. In *International Conference on Computing Education Research*, San Diego, 2013.

[11] D. D. Garcia and D. Ginat. Demystifying computing with magic. In *Special Interest Group on Computer Science Education*, Raleigh, 2012.

[12] R. Taub, M. Ben-Ari, and M. Armoni. The effect of CS Unplugged on middle-school students' view of CS. In *Conference on Innovation and Technology in Computer Science Education*, Paris, 2009.

[13] Exploring Computer Science. Curriculum. URL http://exploringcs.org. Accessed 6 April 2015.

[14] CS Unplugged. About CS Unplugged, 2015. URL http://csunplugged.org. Accessed 9 April 2015.

[15] R. Thies and J. Vahrenhold. Reflections on outreach programs in CS classes: Learning objectives for "unplugged" activities. In *Special Interest Group on Computer Science Education*, Raleigh, 2012.

[16] R. C. Overbaugh and L. Schultz. Bloom's taxonomy. URL http://odu.edu. Accessed 20 March 2015.

[17] Q. Cutts, S. Esper, and B. Simon. Computing as the 4th "R": A general education approach to computing education. In *International Workshop on Computing Education Research*, Rhode Island, 2011.

[18] W. Pohl and V. Dagiene. Bebras international contest on informatics and computer fluency, 2014. URL http://bebras.org. Accessed 12 April 2015.

[19] G. H. Fletcher and J. J. Lu. Human computing skills: Rethinking the K-12 experience. *Communications of the ACM*, 52(2):23–25, 2009.

[20] A. Repenning. Agentsheets: An interactive simulation environment with end-user programmable agents. In *Interaction 2000*, Tokyo, 2000.

[21] A. Repenning and A. Ioannidou. Broadening participation through scalable game design. In *Special Interest Group on Computer Science Education*, Portland, 2008.

[22] A. R. Basawapatna, K. H. Koh, and A. Repenning. Using scalable game design to teach computer science from middle school to graduate school. In *Conference on Innovation and Technology in Computer Science Education*, Bilkent, 2010.

[23] I. Lee, F. Martin, J. Denner, B. Coulter, W. Allan, J. Erickson, J. Malyn-Smith, and L. Werner. Computational thinking for youth in practice. *ACM Inroads*, 2(1):32–37, 2011.

[24] R. J. Mislevy, R. G. Almond, and J. F. Lukas. A brief introducation to evidence-centered design. Technical report, Princeton University, 2003.

[25] D. W. Rutstein, E. Snow, and M. Bienkowski. Computational thinking practices: Analyzing and modeling a critical domain in computer science education. In *American Educational Research Association*, Philadelphia, 2014.

APPENDIX A

ACTIVITY MATRIX - ORIGINS AND ADDITIONS

Table A.1: Matrix showing the origin and revisions to each CS Unplugged activity used in our deployments.

Activity Name	**Origin**	**Refinements**
Computer Vision	CSM	• Created PowerPoint slides detailing real-world applications of computer vision • Developed two sets of worksheets (edge detection, shape recognition)
Routing	CSM	• Developed activity to model a client-server download • Contains some similar concepts to image representation • Created PowerPoint slides to encourage discussion and real-world connections
Finite State Automata (fruit vendor)	CS Unplugged Extension	• Activity found from csunplugged.org link, minor changes to props used (hats => plastic silverware) • Developed "traffic light" guided FSA worksheet

Table A.1: Continued

Binary Numbers	CS Unplugged	• Created a "Check Your Understanding" assessment • Developed "Adding It Up" extension – Students roll dice and add the numbers (in binary) on a scratch piece of paper – Students try to get as close to a certain max value as possible without hitting overflow (5 bits => 31, etc.)
Image Representation	CS Unplugged	• Developed "Rhino MRI" career extension * • Altered existing worksheets to reflect classroom time constraints • Created "Check Your Understanding" assessment
Finite State Automata (treasure map)	CS Unplugged	• Main activity used as-is from csunplugged.org • Developed "Video Game Design" career extension
Artificial Intelligence	CS Unplugged	• Developed real-world connection PowerPoint about Alan Turing • Modernized questions used in the activity • Created "Intelligent Piece of Paper" tic-tac-toe extension • Added career discussion for real-world connection

Table A.1: Continued

Minimal Spanning Trees	CS Unplugged	• Main activity used as-is from csunplugged.org • Augmented with an efficiency extension *
Sorting / Searching	CS Unplugged	• Demonstrated selection / insertion sort in kinesthetic environment • Created "Check Your Understanding" assessment • Added career discussion for real-world connection
Deadlock	CS Unplugged	• Main activity used as-is from csunplugged.org
Cryptology & Information Hiding	CS Unplugged	• Main Information Hiding activity used as-is * • Developed "Criminal Investigation" career extension * • Parity magic trick used as-is
Caesar Cipher and Frequency Analysis	CSM	• Developed cipher worksheets to guide students through encryption • Created frequency analysis data for students to practice decryption • Replaces "Cryptology & Information Hiding" activity

Table A.1: Continued

Parity & Error Correction	CS Unplugged	• Parity magic trick used as-is • Wrote lesson plan to expand magic trick concept to full class period • Designed new worksheets to practice 1D and 2D parity for error detection and correction, respectively

*: This activity was rated poorly by students and is mentioned further in the second table.

Table A.2: Table listing low-ranked activities and possible reasons for their low rankings.

Activity	Refinements	Possible Pain Points
Career Extension: Rhino MRI Scans (Image Representation)	• Made math equations multiple choice to prevent need for a calculator • Added visuals and real-world connection	• Math was too difficult for age level (required students to have a calculator)
Career Extension: Criminal Investigation (Information Hiding)	• Changed average age => average favorite number	• Only one student is engaged at a time • Goals don't make sense... average favorite number?

Table A.2: Continued

Extension: Efficiency in Counting Algorithms (Minimal Spanning Trees)	• Created three different complexity algorithms for counting chairs	• Students are not familiar with mathematical terms • Students / teachers do not fully understand the importance of efficiency after this exercise

APPENDIX B

PRE- AND POST-SURVEY QUESTIONS

[Computing Interest]

Regardless of whether or not you have actually tried it, how interested are you in…

Scale: Very interested = 4, Interested, A Little Interested, Not At All Interested = 1

1. Making computers more intelligent (more like people)?
2. Creating algorithms to make computers faster?
3. Understanding how computers represent data and images?
4. Designing computer games?
5. Computer networking (for example, the internet)?
6. Thinking of new ways to apply computer science (for example, new apps or software)?
7. Programming computers to create new apps (in other words, writing code)?
8. Finding technological solutions to world problems using computer science?

[Computing Confidence]

Right now, how confident are you in your ability to…

Scale: Very Confident = 4, Confident, A Little Confident, Not At All Confident = 1

1. Learn computer science concepts?
2. Make computers more intelligent (more like people)?
3. Create algorithms to make computers faster?
4. Understand how computers represent data and images?
5. Design computer games?
6. Do computer networking?
7. Think of new ways to apply computer science (For example, new apps or software)?
8. Program computers to create new apps (in other words, writing code)?
9. Find technological solutions to world problems using computer science?

[Intent to Persist (computing education and career plans)]

How much would you want to…

Scale: A Lot = 4, Pretty Much, A Little, Not At All = 1

1. Take a computer science class?
2. Get a college degree?
3. Get a computer science or technology-related college degree?
4. Get a computing-related career as an adult?

[Computing Career Knowledge]

I believe that if I were to get a college degree in computing, I would probably…

Scale: I Strongly Agree = 4, I Agree, I Disagree, I Strongly Disagree = 1

1. Make good money
2. Do work that I would enjoy
3. Do work that can “make a difference” in people’s lives
4. Find a job easily

How much do you know about…

Scale: A Lot = 4, Pretty Much, A Little, Not At All = 1

1. Jobs computer scientists have?
2. What computer scientists do in their jobs?

Have you attended a Discovering Technology workshop at the Colorado School of Mines?
Scale: Yes = 1, No = 2

What is your gender?
Scale: Female = 1, Male = 2

What is your race/ethnicity?

African American / Black	1
Native American	2
Asian / Pacific Islander	3
White / Caucasian	4
Hispanic / Latino / Latina	5
Other	6

On a scale of 1 to 5, how much did you enjoy the CS Unplugged activities?
1: I did not enjoy any of the activities
5: I enjoyed every single activity

How hard were the activities?
1: Very Easy (All activities made complete sense)
5: Very Hard (I did not understand the activities)

What did you like BEST about the activities? (optional)
[Paragraph response]

What did you like LEAST about the activities? (optional)
[Paragraph response]

APPENDIX C

CS UNPLUGGED ACTIVITY - COMPUTATIONAL THINKING MATRIX

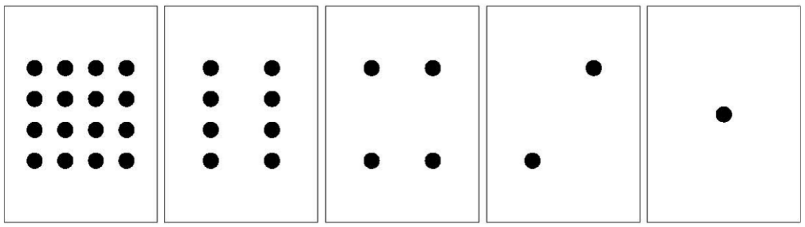
Table C.1: The different Bloom's Taxonomy behaviors present in the CT assessments.

	Data Representation	Decomposition	Pattern Recognition	Pattern Generalization & Abstraction	Algorithm Design
Artificial Intelligence		X			
Binary Numbers	X				
Caesar Cipher and Frequency Analysis					X
Computer Vision	X			X	
Deadlock and Routing				X	
Finite State Automata			X		
Image Representation	X				
Minimal Spanning Trees					X
Parity and Error Correction		X	X		
Sorting and Searching					X

APPENDIX D

BINARY NUMBERS ASSESSMENT

Name: _____ **Check Your Understanding**



Count the Dots – Binary Numbers

1. What is the next number in the sequence?
   00001 00010 00011 00100 _____

2. What decimal number is represented by 01011?

3. How would you write the number 20 in binary?

4. What is the largest number you can represent using five cards (i.e., five bits)?

5. What is the largest number you could represent if you had only three cards?

6. How many cards (bits) would you need to represent the number 63?

APPENDIX E

BINARY NUMBERS RUBRIC

Table E.1: Rubric for Binary Numbers Worksheet

	3 Proficient	2 Partially Proficient	1 Unsatisfactory
What is the next number in the sequence?	Student correctly identifies the pattern and answers 00101 (five).	Student correctly identifies the answer should be in binary, but does not recognize the pattern and gives an incorrect number.	Student didn't attempt the problem or answered in decimal numbers.
What decimal number is represented by 01011?	Student converts from binary to decimal and answers with 11.	Student converts from binary to decimal, but gives an incorrect answer (such as re-converting 11 to decimal number 3).	Student does not convert the number to a decimal representation, or simply adds a decimal point to the binary number.
How would you write the number 20 in binary?	Student correctly answers 10100.	Student answers incorrectly but gives some level of justification explaining their reasoning.	Student does not attempt the problem or gives an answer using decimal numbers.
What is the largest number you can represent using five cards (i.e., five bits)?	Student answers 31.	Student gives an incorrect answer but with justification behind their thought process.	Student does not attempt the problem or gives an answer without justification.

Table E.1: Continued

What is the largest number you could represent if you had only three cards?	Student answers 7.	Student answers 28 (the highest of the five bit cards on the worksheet) or another incorrect answer, but gives justification.	Student answers incorrectly and without justification.
How many cards (bits) would you need to represent the number 63?	Student answers 6.	Student answers 7 bits (off by one error).	Student answers 5 or fewer bits (this should be obviously wrong with the bit cards printed at the top), or another number without justification.

APPENDIX F

CRYPTOLOGY ASSESSMENT

Worksheet – The Caesar Cipher

Julius Caesar used a simple substitution cipher to send messages to his troops. He used a very simple rule to replace each letter with another letter from the alphabet. He substituted each letter by the letter that was 3 places further along in the alphabet, so that “a” was replaced with “D”, “b” with “E” and so on.

Complete the table below to show what each letter is enciphered as using this system.

a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
D	E	F																							

Using the Caesar Cipher, encode the name of your school. Check that you get the same code as the person sitting next to you.

_____

Cut out and make a Caesar cipher wheel. Use the wheel to encipher your name using two different keys. Pass your encoded names to the person next to you, and ask them to work out what keys you used.

How many different keys are there? _____

Decode this message, which was encoded using the Caesar cipher from the table above:

Z	K	H	Q		B	R	X		K	D	Y	H		G	H	F	R	G	H	G		W	K	L	V

Z	R	U	N		R	X	W		W	Z	H	Q	W	B		V	H	Y	H	Q		W	L	P	H	V

Q	L	Q	H		D	Q	G		W	H	O	O		B	R	X	U		W	H	D	F	K	H	U

Write a short message of your own (three or four words), and encipher it using the Caesar cipher wheel. Hand the secret message to the person sitting next to you, and have them decipher it.

Material from http://crypto.interactive-maths.com/

Worksheet – Frequency Analysis

Using what you have learned, try to decrypt the text below. Look for letters which are frequently used and short words which might be easy to decode.

DXWG GXJQ KNF'G CFKJLSGWFK WPNCG PHLGXKWQS WFK DXWG GXJQ FJUJL GJII QNC HS GXWG DXJF QNC'LJ JIJUJF, QNC'LJ WISN GJF, WFK FHFJ, WFK JHYXG, WFK SJUJF, WFK SHT, WFK ZHUJ, WFK ZNCL, WFK GXLJJ, WFK GDN, WFK NFJ.

WFK DXJF QNC DWBJ CR NF QNCL JIJUJFGX PHLGXKWQ QNC JTRJOG GN ZJJI JIJUJF, PCG QNC KNF'G. QNC NRJF QNCL JQJS WFK JUJLQGXHFY'S ECSG IHBJ QJSGJLKWQ, NFIQ HG'S GNKWQ. WFK QNC KNF'G ZJJI JIJUJF WG WII.

QNC ZJJI IHBJ QNC'LJ SGHII GJF. WFK QNC WLJ - CFKJLFJWGX GXJ QJWL GXWG AWBJS QNC JIJUJF.

APPENDIX G

CRYPTOLOGY RUBRIC

Table G.1: Rubric for Cryptology Worksheets

	3 Proficient	2 Partially Proficient	1 Unsatisfactory
Caesar Ciphers: Encrypting a Plaintext Message	Student is able to complete an existing cipher to encode a plaintext value correctly.	Student can complete a partial Caesar cipher, but is unable to encrypt a plaintext message or inconsistently encrypts data (top-to-bottom *and* bottom-to-top encryption).	Student is unable to demonstrate knowledge of a Caesar cipher, and cannot complete a partial cipher nor use a cipher to encrypt a plaintext message.
Caesar Ciphers: Analyzing the Number of Unique Cipher Keys	Student acknowledges 25 or 26 possible Caesar ciphers based on the in-class cipher.	Student recognizes a number of Caesar ciphers that can be justified using the number of letters in the alphabet (e.g., 26*26, 26*25, and so on).	Student analyzes the number of possible Caesar cipher keys and answers with a number unrelated to the number of letters in the alphabet.
Caesar Ciphers: Decryption of a Ciphertext Message Using a Known Cipher	Student is able to take an encrypted message and a known cipher key to produce a plaintext message.	Student takes an encrypted message and a known cipher to produce a doubly encrypted message.	Student is unable to connect an existing cipher with an encrypted message, and does not produce a message.

Table G.1: Continued

Substitution Ciphers: Applying Frequency Analysis to Decrypt a Ciphertext Message	Student demonstrates some of the following ideas discussed in class: counting the frequency of letters and matching with the English frequency chart, making intelligent guesses on an encrypted word based on the length of the word. Several letters have been guessed in the ciphertext message.	Student demonstrates some understanding of frequency analysis using those listed in the proficient category, but are only able to guess one letter in the encrypted text.	Student does not attempt the decryption and guesses at 0 letters.

APPENDIX H

ERROR DETECTION AND PARITY ASSESSMENT

ASCII: Storing Letters as Numbers (1D)

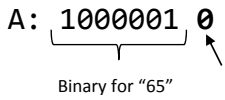
ASCII stands for the American Standard Code for Information Interchange. It's a system used to represent English characters, and it was designed to encode 128 different characters. Seven bits are used to store the ASCII encoding of the character, while an eighth bit is used for *parity* error detection. Below is part of the ASCII table (the part that shows capital letters):

A 1000 001	B 1000 010	C 1000 011	D 1000 100	E 1000 101	F 1000 110
G 1000 111	H 1001 000	I 1001 001	J 1001 010	K 1001 011	L 1001 100
M 1001 101	N 1001 110	O 1001 111	P 1010 000	Q 1010 001	R 1010 010
S 1010 011	T 1010 100	U 1010 101	V 1010 110	W 1010 111	X 1011 000
Y 1011 001	Z 1011 010				

First, let's try translating this message from binary numbers to English letters:

1000 010 1000 101 1000 111 1001 001 1001 110

When saving data to your computer or sending data over the internet, errors can happen. The character "A" is the number 65 in binary. The number 65 only takes seven bits to represent, and the eighth bit is used as a *parity* bit to try and detect if an error happened while saving the letter to your computer.



Fill in the parity bits in the above ASCII table. A parity bit is 0 if there are an *even* number of 1's in the binary number, or it is 1 if there are an *odd* number of 1's in the binary number.

Below is the same message, but this time it was sent with parity bits. Is there an error in the message? **Circle a binary number if you think it was sent incorrectly.**

1000 0100 1000 1011 1000 1110 1001 0010 1001 1100

If an ASCII number is sent in error (it doesn't have an even number of 1's), can you figure out what letter was *supposed* to be sent? (aka Can you correct the message?) Why or why not?

_____

_____

If you finish early, try writing a message using ASCII below.

Error Detection and Correction (2D)

Computers store all data as binary numbers – even the letters A through Z. You may have made a spelling error before in an email or when writing a paper. Sometimes, computers make mistakes when sending data between each other.

Error Detection

We have just received a message. Before we convert the binary numbers to letters, we need to check if the message contains any errors by checking the *parity bits*. The rightmost column and bottom row are parity bits. Using even parity, circle the row and column of any errors. **Can you tell which number has an error?**

0 0 0 1 0	0 0 0 0 0	1 0 0 0 1	0
0 0 0 1 1	0 1 1 1 0	0 0 1 1 0	1
0 0 0 0 1	0 1 0 0 0	1 0 0 1 1	1
0 0 0 0 0	0 0 1 1 0	0 0 1 1 0	0

Error Correction

After realizing there is an error, you ask the computer to resend the number with the error. The computer responds back with "1 0 0 1 1". Write this number above the erroneous number and recheck the parity bits.

If you're confident that there are no more errors in the message, let's decode it and see what it says. Use the following conversion table to convert the ***message*** (do not convert the parity row or parity column). Write down the words next to the rows.

Conversion Chart

A: 00000	E: 00100	I: 01000	M: 01100	Q: 10000	U: 10100	
B: 00001	F: 00101	J: 01001	N: 01101	R: 10001	V: 10101	Y: 11000
C: 00010	G: 00110	K: 01010	O: 01110	S: 10010	W: 10110	Z: 11001
D: 00011	H: 00111	L: 01011	P: 01111	T: 10011	X: 10111	

We now want to respond to the computer's message with the one below. Before we send it, we need to add parity bits so that the other computer can check for errors. **Add parity bits in the empty column and row below.**

1 0 0 1 1	0 0 0 0 0	0 0 1 1 0	
0 1 1 0 0	1 0 1 0 0	0 0 1 1 0	
1 0 0 0 1	0 1 1 1 0	1 0 0 1 1	

APPENDIX I

ERROR DETECTION AND PARITY RUBRIC

Table I.1: Rubric for Error Detection Worksheets

	3 Proficient	2 Partially Proficient	1 Unsatisfactory
Data Representation	Student uses the given letter mapping and translates from binary numbers to characters with no errors.	Student uses the given letter mapping, and converts most numbers correctly (only one error).	Student is unable to decode 7-bit ASCII values using a given ASCII table and message.
1D Parity Bits	Student correctly computes the parity bits for letters in teh ASCII table.	Student partially completes the parity bits for the ASCII table, or has some incorrect parity bits.	Student does not attempt to add parity bits to the ASCII table.
1D Error Detection	Student correctly identifies the fourth letter to have been sent with an error, and is able to justify why it cannot be corrected.	Student identifies two or more letters containing an error, and is unable to justify why the error cannot be corrected. OR Student is able to justify why the error cannot be corrected, but does not identify the correct letter.	Student identifies no letters containing an error (or did not attempt the problem).

Table I.1: Continued

2D Parity Bits	Student correctly computes the parity bits for letters in the message.	Student partially completes the parity bits for the message, or has some incorrect parity bits.	Student does not attempt to add parity bits to the message.
2D Error Detection	Student correctly identifies only one error, and is able to correct the error by using the intersection of the row and column.	Student identifies two or more errors, or attempts to correct an error by changing bits located outside of the intersection.	Student does not attempt to identify an error or correct an error in the 2D grid.

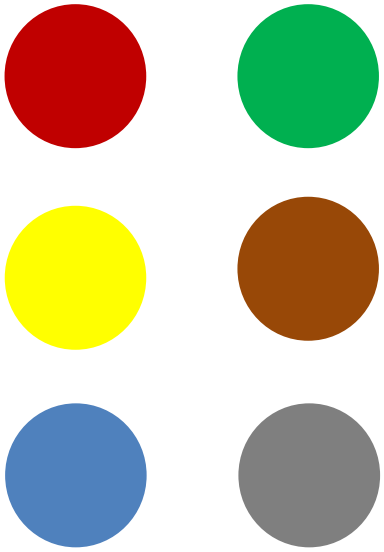
APPENDIX J

SORTING AND SEARCHING ASSESSMENT

Secret Weights - Don't show your partners!

The number represents the "weight" of each color. (Ex: Yellow is "lighter" than Red because 3 is less than 44)

RED = 44 GREEN = 88 YELLOW = 3 BROWN = 81 BLUE = 92 GRAY = 14



Sorting Colors!

You have been given 6 Color Cards with a different color on each card. Each color has a different "weight". Follow the instructions to sort the colors from "lightest" to "heaviest" by asking **yes or no** questions about **two** colors at a time. You will end up sorting weights without actually knowing them!

1. Lay the Color Cards down in the following order: Red, Green, Yellow, Brown, Blue, and Gray. This is not the correct order. Remember, you are trying to ask questions about the colors to put them in the correct order.
2. Ask "yes" or "no" questions to try and find out where each color will go in the "lightest" to "heaviest". (*Ex: Is Red lighter than Green?)*
3. Record your questions and answers to keep track of your work.
4. Move the cards around as you find out more information about them making sure to end up with a "lightest" to "heaviest" line-up. Once you are certain a card is in the correct spot, flip that card over (face-down).
5. All cards that are face-up are *unsorted.* As cards are flipped face-down, they become *sorted.*
6. When all cards are face-down, you have sorted your colors! Compare with your partner to see if you are correct.

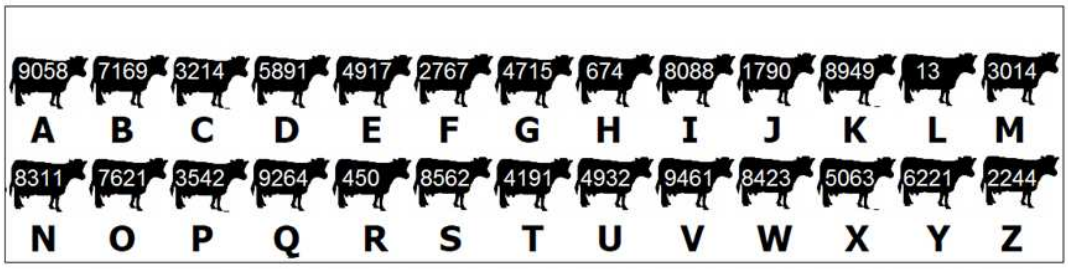
Record your questions and answers. If you need more space, use the back of this sheet of paper.

Write your question:	Circle Answer:
1. _____	Yes/No
2. _____	Yes/No
3. _____	Yes/No
4. _____	Yes/No
5. _____	Yes/No
6. _____	Yes/No
7. _____	Yes/No
8. _____	Yes/No
9. _____	Yes/No
10. _____	Yes/No

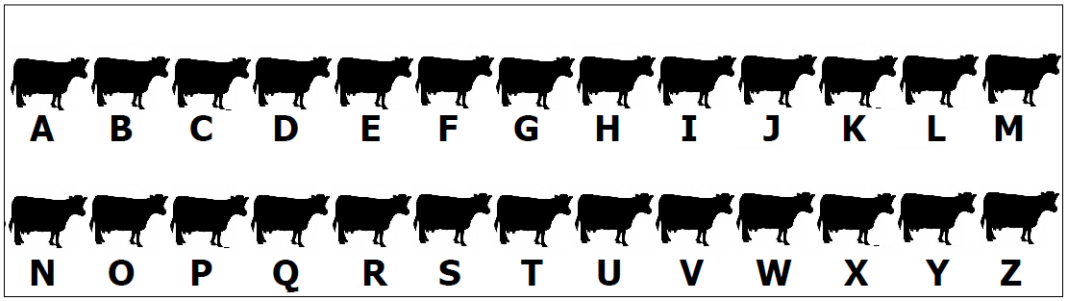
Write down the colors in their *sorted* positions here:

_____ _____ _____ _____ _____ _____

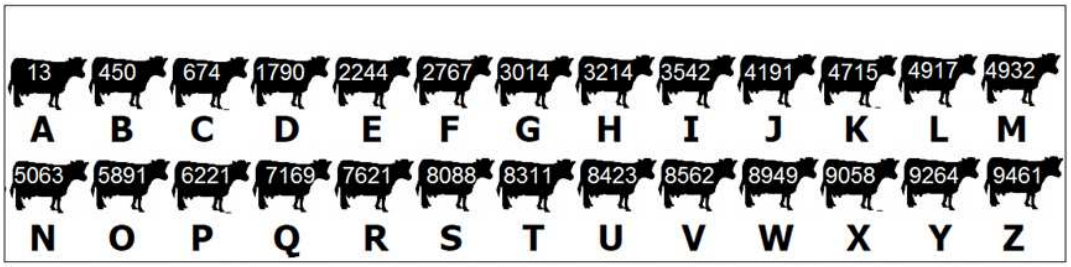
Kingdom 1 Records Number of Guesses:



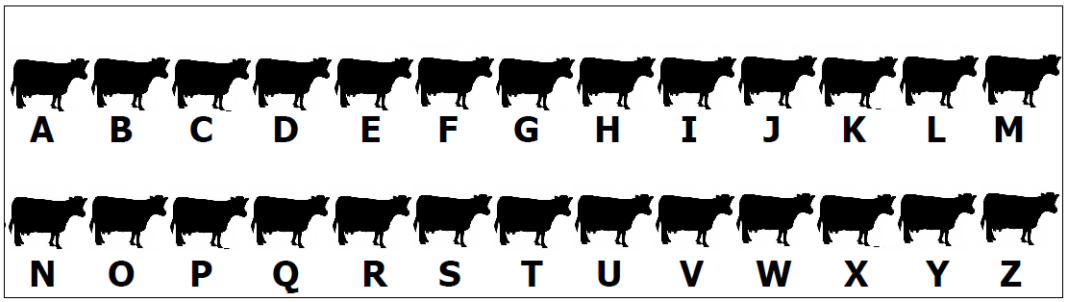
Your Records Number of Guesses:



Kingdom 2 Records Number of Guesses:



Your Records Number of Guesses:



APPENDIX K

SORTING AND SEARCHING RUBRIC

Table K.1: Rubric for Sorting and Searching Worksheets

	3 Proficient	2 Partially Proficient	1 Unsatisfactory
Application (Do students use binary/linear searches when given a searching problem?)	Student uses an appropriate searching algorithm (linear or binary search) to search for the cow in question. Cows are marked on the worksheet to show which ones have been queried. Random search will be accepted as Proficient *only* for students who were searching through unsorted cows.	Student uses a random search to locate the cow (if they could have used a binary or linear search). OR Student uses binary search to try and search through unsorted data.	Student does not search for the cow or does not mark the cows on the searching worksheet.

Table K.1: Continued

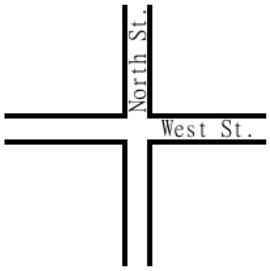
Application (Do students use quicksort/selection sort when given a sorting problem?)	Student uses an appropriate sorting algorithm (quicksort, selection sort) to discover the order of the colored circles. The sorting algorithm can be determined by well defined yes/no questions on the worksheet. Answer is unimportant.	Student uses seemingly random questions to ascertain the order of the colored discs, and questions are well defined and logged in the worksheet. Answer is unimportant.	Student does not log the questions their group asked or the order of the colored circles differes from the answers to the questions they wrote down.

APPENDIX L

FSA ASSESSMENT

Worksheet: Traffic Lights

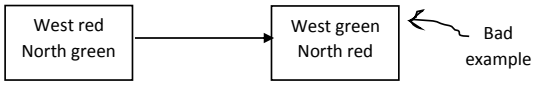
A traffic light is a real world example of something that can be modeled as a finite-state automaton. The city of Villagetown is not familiar with how traffic lights work, so they have tasked you with designing a traffic light for the intersection of North Street and West Street, pictured to the right:



You want to make sure that the traffic lights behave correctly, so you have decided to model them with a finite state diagram. First, you need to figure out which states you need. You have determined that there are nine possible states, but not all of them make sense. **Circle the states you actually need to use (states that would *not* cause an accident):**

West green (WG) North green (NG)	West yellow (WY) North green (NG)	West red (WR) North green (NG)
West green (WG) North yellow (NY)	West yellow (WY) North yellow (NY)	West red (WR) North yellow (NY)
West green (WG) North red (NR)	West yellow (WY) North red (NR)	West red (WR) North red (NR)

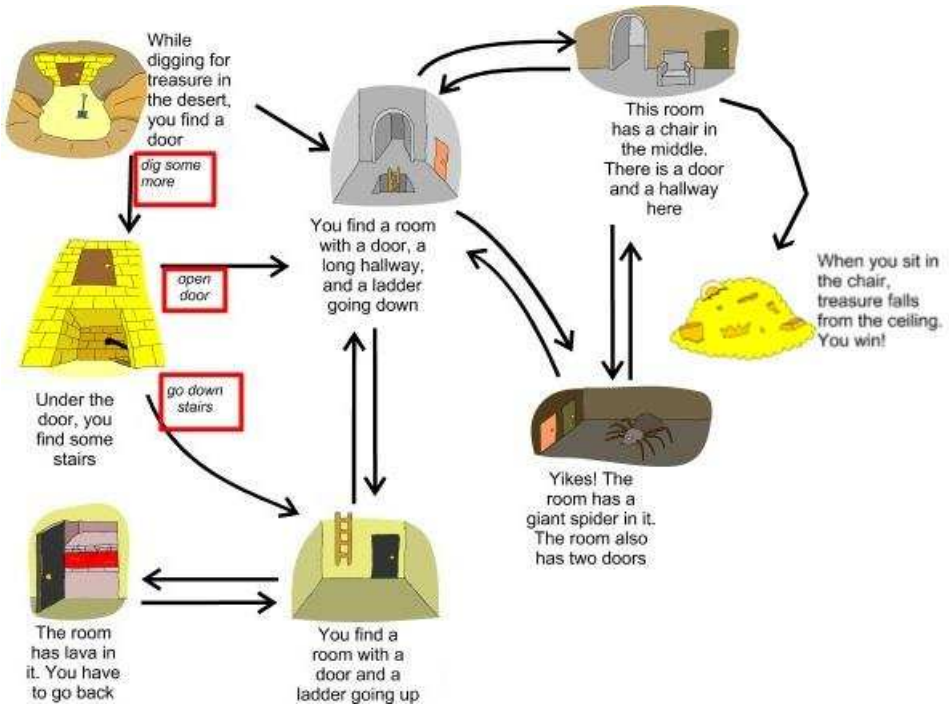
Now, using the states you circled, draw a finite state diagram for the traffic lights in the box on the bottom of this worksheet. Be careful about the transitions you draw, because you don't want to cause accidents. For example, the below transition might not be a good one. Can you see why? **Which state should be your start state?**



Finite State Automata: Check Your Understanding

Below is part of a finite state automata diagram.

1. Fill in the missing transitions (3 have been given to you – they are marked in boxes)
2. Determine which state is the ***start*** state (and mark it!)
3. Draw double circles around the final (winning) state
4. Figure out at least two different paths to the treasure!



Can you think of anything else in the real world that can be modeled with a Finite State Automata?

_____

APPENDIX M

FSA RUBRIC

Table M.1: Rubric for FSA Worksheets

	3 Proficient	2 Partially Proficient	1 Unsatisfactory
Traffic Light State Selection	Student identifies four or five traffic signal states that will not cause an accident.	Student identifies at least two states that will not cause accidents (possibly borrowing the two states from the bad example), and may identify some states that will cause accidents.	Student identifies no states that cause no accidents, and/or identifies states that will cause accidents.
Finite State Automata Construction	Student is able to represent a traffic signal using FSA symbols (states, transitions, start, stop). Transitions clearly show direction.	Student is able to partially model a traffic signal using FSA symbols. Direction of transition is not clear.	Student is unable to represent the relationship between the selected states using transitions.
Fill in the Transition	Student is able to clearly identify the descriptor missing from the transition arrow by using examples and context from the states.	Student is able to add some transition descriptors, but not all descriptors make sense.	Student does not add any descriptors or adds new transitions that were not present in the original file.

APPENDIX N

WORKSHEET SUMMARY TABLES

Table N.1: Abbreviations of CT Skills.

Data Representation	DR
Decomposition	D
Pattern Recognition	PR
Pattern Generalization & Abstraction	A
Algorithmic Thinking	ALG

Table N.2: Binary Numbers Worksheet Review

	Question	Description	Bloom's	CT	p
Q1	What is the next number in the sequence? 00001 00010 00011 00100 _____	This question tests conversion between number systems to recognize patterns.	Apply	DR, PR	
Q2	What decimal number is represented by 01011?	Individually tests conversion from binary to decimal number systems.	Remember / Understand	DR, PR	
Q3	How would you write the number 20 in binary?	Individually tests conversion from decimal to binary number systems.	Remember / Understand	DR	
Q4	What is the largest number you can represent using five bits?	Assesses understanding of number places in binary numbers.	Analyze	DR, A	
Q5	What is the largest number you could represent if you had only three cards?	Assesses ability to generalize number places from examples using five bits and apply to a question about three bits.	Analyze	PR	
Q6	How many cards (bits) would you need to represent the number 63?	Assesses knowledge of number places and decimal to binary conversion.	Analyze	DR, A	*

Table N.3: Cryptology Worksheets Review

	Description	Bloom's	CT	p
Encryption	Using a cipher, students create a ciphertext message of their school's name.	Applying	PR	
Analysis	Analysis of the Caesar cipher wheel and the number of unique cipher keys.	Analyze	PR	
Decryption	Using a cipher, students extract a plaintext message from an encrypted message.	Applying	ALG	
Application	Students apply frequency analysis techniques to begin decoding a message encrypted with a substitution cipher.	Evaluating	PR, D	

Table N.4: Error Detection Worksheets Review

	Description	Bloom's	CT	p
Data Representation	Use ASCII to interpret binary numbers as letters.	Under-standing	ALG	
Parity Bits (1D)	Using even parity, add a parity bit to the 7-bit ASCII values.	Applying	ALG	
Error Detection (1D)	Using even parity, detect if any numbers were transmitted incorrectly.	Evaluate	ALG	
Error Detection (2D)	Using even parity, detect if any numbers were transmitted incorrectly and correct.	Analyze	ALG, D	
Parity Bits (2D)	Using even parity, add parity bits to a message.	Applying	ALG, D	

Table N.5: Searching Worksheet Review

	Description	Bloom's	CT	p
Application	Given a list of unsorted or sorted numbers, search for a known value.	Evaluating	ALG, D	

Table N.6: Sorting Worksheet Review

	Description	Bloom's	CT	p
Application	Sort colors based on hidden knowledge known by your partner.	Evaluating	ALG, A, D	

Table N.7: FSA Worksheet Review

	Description	Bloom's	CT
State Selection	Given all possible states for a traffic light, choose states that do not cause an accident.	Evalute	A
FSA Construction	Using states chosen in the prior step, connect them using conventions learned in class.	Creating	A
Transitions	Given states and transitions, fill in the descriptor needed to move between states.	Under-standing, Analyzing	A

APPENDIX O

FINAL PROJECT (PET VERSION)

Name: _____

Pet Mystery

Do you ever wonder how your pets spend their day while you are at school? Maybe they are actually experts in computer science…

Collect clues from different animals by completing their challenge. Then, pair the clues together to see if you can uncover a secret message.

Clue #1

(This is your free clue – Complete other challenges to get more)
Each animal has their weight next to their name.


Fuzzy, 15


Guzzy, 12


Nuzzy, 5


Harry, 24


Larry, 28


Kerry, 35


Perry, 31


Wuzzy, 50


Buzzy, 23


Barry, 42

It might be useful to write down the animals from lightest to heaviest.

_____

What process did you use to put them in order?

_____

Name: _____

Detective Notes: Numbers and Symbols

Did you know that every number between 0 and 32 can be written as a sum of the following numbers: 1, 2, 4, 8, and 16?

Example: The number 6 can be written as 0 + 0 + 4 + 2 + 0
Note that 0's were added instead of the numbers 16, 8, and 1.

The alphabet is listed on the right side of this page, and each letter is given a corresponding number.

Challenge 1: Can we figure out how to decode the following message?

	16	8	4	2	1
	0	0	1	1	0
	1	0	0	1	0
	0	1	0	0	1
	0	0	1	0	1
	0	1	1	1	0
	0	0	1	0	0

Challenge 2:

Your dog has buried 100 shoes in a straight line in the back yard. Because your dog is very organized, she buried the shoes according to their shoe size. You want to find a particular shoe, and you can see the mounds of dirt where all the shoes are buried. What is the maximum number of shoes you need to dig up to find the shoe you're looking for?

Write your answer here: _____

Below, we've listed each letter and its numerical place in the alphabet.

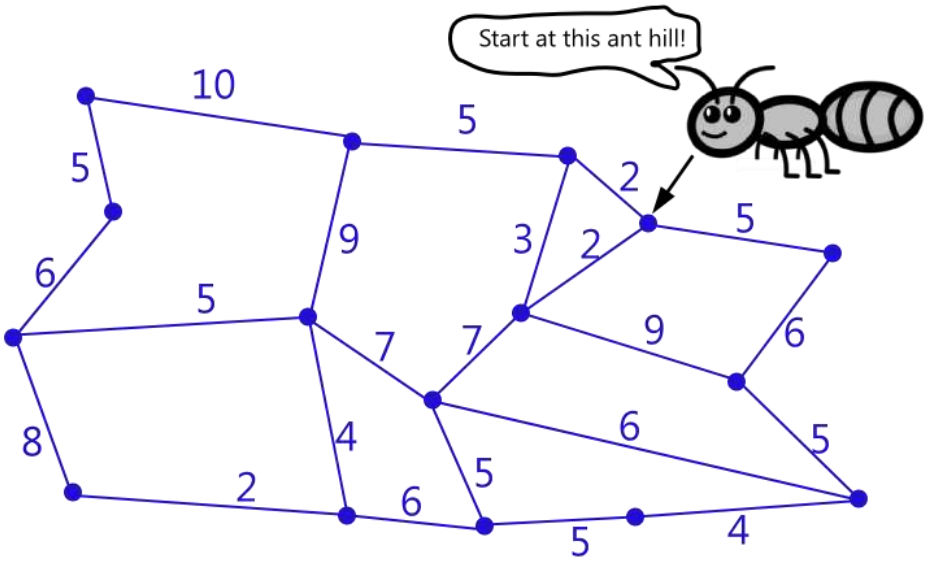
A = 1
B = 2
C = 3
D = 4
E = 5
F = 6
G = 7
H = 8
I = 9
J = 10
K = 11
L = 12
M = 13
N = 14
O = 15
P = 16
Q = 17
R = 18
S = 19
T = 20
U = 21
V = 22
W = 23
X = 24
Y = 25
Z = 26

Name: _____

Detective Notes: Ants

Each dot in the graph below represents a different ant hill. All the queen ants have decided they should connect their ant hills with tunnels. The numbers next to each line represent the distance, in feet, between the ant hills, and all the possible ways the ant hills *could* connect.

Shade the lines where the ants should dig tunnels so that the ants have to dig the shortest amount of tunnels to connect every ant hill. Once complete, you will be given your next clue.



How did you decide which tunnels to dig?

_____

_____

_____

Name: _____

Detective Notes: Dog

Delilah the dog has an exhausting schedule. She's got to remember to bark at the mailman, run around the yard, and take plenty of naps. This particular dog follows a set of rules in her schedule.

The Rules (the events are in bold):

1. **Naps** only happen after **treats** are given out or after **barking at the mailman**
2. **Playing fetch** can be done after a **treat** or after **barking at the mailman**
3. **Barking at the mailman** only happens after a **nap**
4. **Burying a bone** must be the last thing the dog does for the day
5. The dog will **bury a bone** only after she has **barked at the mailman** or she has **wagged her tail**
6. If she **wags her tail**, it will be after she **plays fetch**

Using the blank space below, organize the events and transitions so it is easy to see the dog's possible schedules.

Circle any of the following schedules if they follow the dog's rules. Additionally, **use the bottom of the page to write another schedule that would work.**

1. Play fetch
2. Nap
3. Bark at the mailman
4. Bury a bone

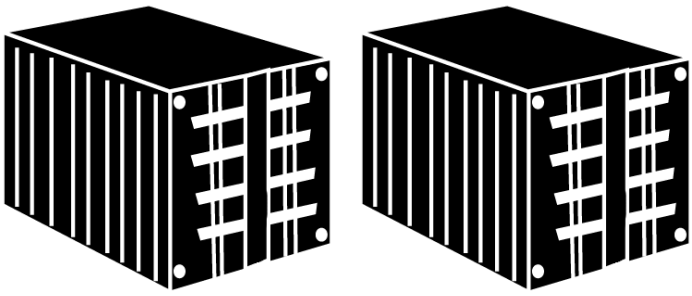
1. Get a treat
2. Nap
3. Bark at the mailman
4. Bury a bone

1. Get a treat
2. Play fetch
3. Wag tail
4. Bury a bone

Name: _____

Detective Notes: Vet Visit

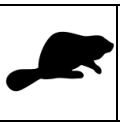
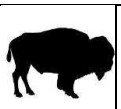
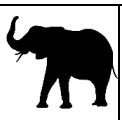
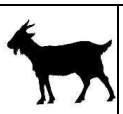
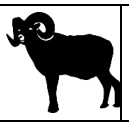
When you take your pets to the vet, you may have used an animal carrier to put them in the car. Vets do the same thing with really large animals. Right now, there are two containers that are empty (no animals inside), and each container can hold a certain amount of weight.



Carrier 1 Carrier 2

The vet needs to load as many animals into these two containers as possible (buying extra containers is expensive). Each container can hold only 300 pounds.

Which animals below should the vet put into the containers? **Beneath each animal, write the number of the container it should go into. The vet won't be able to fit every animal, so some animals will not have a container number.**

							
Weight	**80**	**120**	**80**	**230**	**90**	**110**	**50**
Container							

How did you choose the *first* animal to put into the animal carrier?

_____

_____

Name: _____

Clue #2 (Ants)

Here is a cipher for *encoding* messages.

A	→	A
B	→	D
C	→	F
D	→	G
E	→	E
F	→	H
G	→	J
H	→	K
I	→	I
J	→	L
K	→	M
L	→	N
M	→	P
N	→	Q
O	→	O
P	→	R
Q	→	S
R	→	T
S	→	V
T	→	W
U	→	U
V	→	X
W	→	Y
X	→	Z
Y	→	B
Z	→	C

Clue #3 (Dog)

We know that **two** different symbols are used. Your dog added a parity row/column in case someone messed up the message before you received it.

O	X	O	X	O	X
X	O	X	O	X	X
O	O	O	X	X	O
O	O	O	X	X	O
O	O	O	X	O	X
X	X	X	X	X	X

Clue #4 (Vet)

3

APPENDIX P

FINAL PROJECT (PET VERSION) ANSWER KEY

Name: _____

Pet Mystery

Do you ever wonder how your pets spend their day while you are at school? Maybe they are actually experts in computer science…

Collect clues from different animals by completing their challenge. Then, pair the clues together to see if you can uncover a secret message.

Clue #1


Fuzzy


Guzzy


Nuzzy


Harry


Larry


Kerry


Perry


Wuzzy


Buzzy


Barry

Your crazy neighbor named all of their animals so that the names would rhyme. He plays favorites, and told you the following:

Fuzzy, Guzzy, and Nuzzy are all less than Buzzy.

Fuzzy is better than Guzzy.

Nuzzy is less than Guzzy.

Note that we are given information to "sort" four of the ten animals. We can order as Nuzzy, Guzzy, Fuzzy, Buzzy, {everyone else}

Use Clue #4 (the number 3) to select the third index: Fuzzy.

Name: _____

Detective Notes: Numbers and Symbols

Did you know that every number between 0 and 32 can be written as a sum of the following numbers: 1, 2, 4, 8, and 16?

Example: The number 25 can be written as 16 + 8 + 0 + 0 + 1
Note that 0's were added instead of the numbers 4 and 2.

The alphabet is listed on the right side of this page, and each letter is given a corresponding number.

Challenge 1: Can we figure out how to decode the following message?

	16	8	4	2	1
F	0	0	1	1	0
R	1	0	0	1	0
I	0	1	0	0	1
E	0	0	1	0	1
N	0	1	1	1	0
D	0	0	1	0	0

Challenge 2:

Your dog has buried 100 shoes in a straight line in the back yard. Because your dog is very organized, she buried the shoes according to their shoe size. You want to find a particular shoe, and you can see the mounds of dirt where all the shoes are buried. What is the maximum number of shoes you need to dig up to find the shoe you're looking for?

Write your answer here: _____

100/2 = 50; 50/2 = 25; 25/2 = 13; 13/2 = 7; 7/2 = 4; 4/2 = 2; 2/2 = 1
We need to divide the shoes by two 7 times to get down to one item.

Below, we've listed each letter and its numerical place in the alphabet.

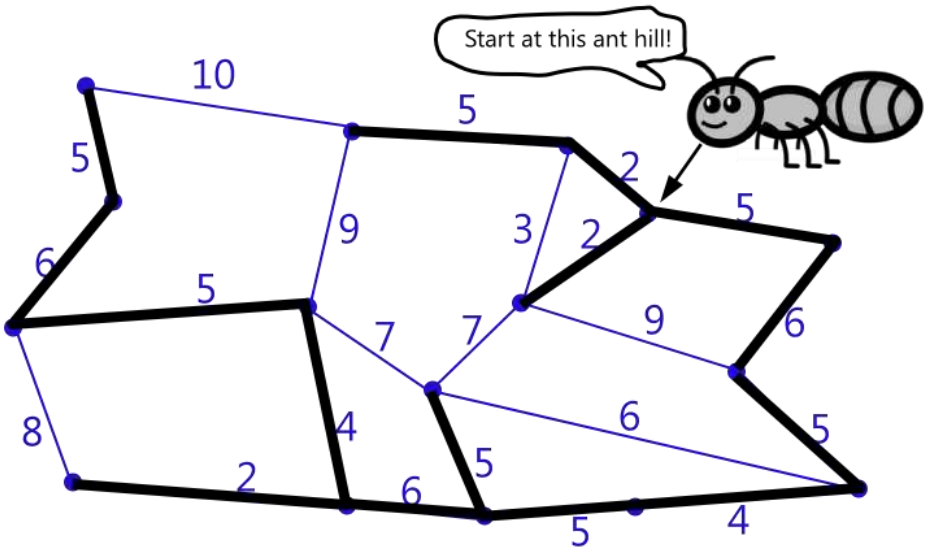
A = 1
B = 2
C = 3
D = 4
E = 5
F = 6
G = 7
H = 8
I = 9
J = 10
K = 11
L = 12
M = 13
N = 14
O = 15
P = 16
Q = 17
R = 18
S = 19
T = 20
U = 21
V = 22
W = 23
X = 24
Y = 25
Z = 26

Name: _____

Detective Notes: Ants

Each dot in the graph below represents a different ant hill. All the queen ants have decided they should connect their ant hills with tunnels. The numbers next to each line represent the distance, in feet, between the ant hills, and all the possible ways the ant hills *could* connect.

Shade the lines where the ants should dig tunnels so that the ants have to dig the shortest amount of tunnels to connect every ant hill. Once complete, you will be given your next clue.



How did you decide which tunnels to dig?

This problem can be solved in two ways:

The hard way (pre-CS Unplugged): try to intelligently shade paths, constantly adding up the total length of the tunnels until you think you have a good solution.

The easy way (post-CS Unplugged): use Kruskal's algorithm to construct a minimal spanning tree. Start by adding the cheapest tunnels (of length 2), before adding those of length 3, 4, 5, and so on until all ant hills are connected.

_____

Name: _____

Detective Notes: Dog

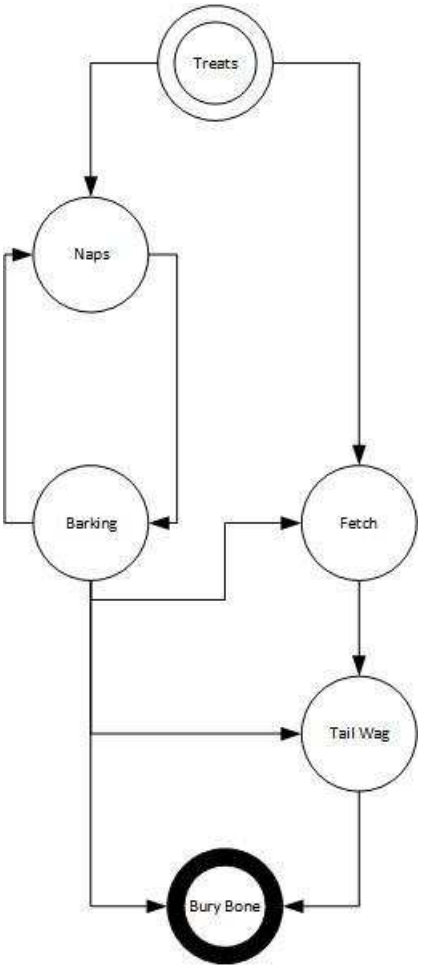
Delilah the dog has an exhausting schedule. She's got to remember to bark at the mailman, run around the yard, and take plenty of naps. This particular dog follows a set of rules in her schedule.

The Rules (the events are in bold):

1. **Naps** only happen after **treats** are given out or after **barking at the mailman**
2. **Playing fetch** can be done after a **treat** or after **barking at the mailman**
3. **Barking at the mailman** only happens after a **nap**
4. **Burying a bone** must be the last thing the dog does for the day
5. The dog will **bury a bone** only after she has **barked at the mailman** or she has **wagged her tail**
6. If she **wags her tail**, it will be after she **plays fetch**

Using the blank space below, organize the events and transitions so it is easy to see the dog's possible schedules.

Name: _____



Circle any of the following schedules if they follow the dog's rules. Additionally, **use the bottom of the page to write another schedule that would work**

The second and third columns are valid schedules.

1. Play fetch
2. Nap
3. Bark at the mailman
4. Bury a bone

1. Get a treat
2. Nap
3. Bark at the mailman

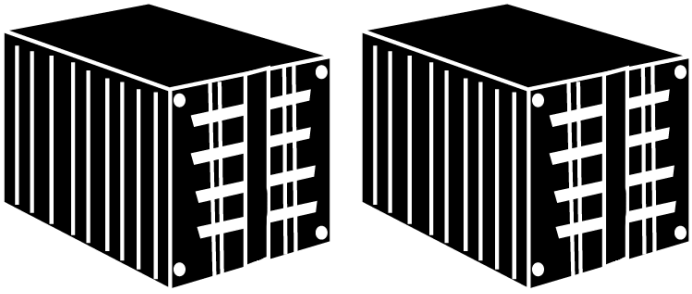
1. Get a treat
2. Play fetch
3. Wag tail

[Answers may vary]

Name: _____

Detective Notes: Vet Visit

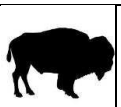
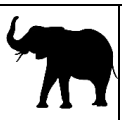
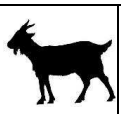
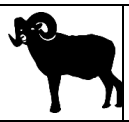
When you take your pets to the vet, you may have used an animal carrier to put them in the car. Vets do the same thing with really large animals. Right now, there are two containers that are empty (no animals inside), and each container can hold a certain amount of weight.



Carrier 1 Carrier 2

The vet needs to load as many animals into these two containers as possible (buying extra containers is expensive). Each container can hold only 300 pounds.

Which animals below should the vet put into the containers? **Beneath each animal, write the number of the container it should go into. The vet won't be able to fit every animal, so some animals will not have a container number.**

							
Weight	**80**	**120**	**80**	**230**	**90**	**110**	**50**
Container	**1**	**1**		**2**	**1**		**2**

How did you choose the *first* animal to put into the animal carrier?

To find the optimal solution, you want to maximize the weight in each container. While the largest animal (elephant) is indeed part of the solution, students will not arrive at the maximized distribution of weight if they try to fit both of the next two heaviest animals.

Container 1: Beaver + Buffalo + Goat = 290;

Container 2: Elephant + snake = 280; you cannot get more weight in a container

_____

_____

Name: _____

Clue #2 (Cat)

Here is a cipher for *encoding* messages.

A	→	A
B	→	D
C	→	F
D	→	G
E	→	E
F	→	H
G	→	J
H	→	K
I	→	I
J	→	L
K	→	M
L	→	N
M	→	P
N	→	Q
O	→	O
P	→	R
Q	→	S
R	→	T
S	→	V
T	→	W
U	→	U
V	→	X
W	→	Y
X	→	Z
Y	→	B
Z	→	C

Clue #3 (Dog)

We know that **two** different symbols are used. Your dog added a parity row/column in case someone messed up the message before you received it.

O	X	O	X	O	X
X	O	X	O	X	X
O	O	O	X	X	O
O	O	O	X	X	O
O	O	O	X	O	X
X	X	X	X	X	X

O	X	O	**O**	O	X
X	O	X	O	X	X
O	O	O	X	X	O
O	O	O	X	X	O
O	O	O	X	O	X
X	X	X	X	X	X

Correct error using parity bits.

O	X	O	O	O	01000	8	H
X	O	X	O	X	10101	21	U
O	O	O	X	X	00011	3	C
O	O	O	X	X	00011	3	C
O	O	O	X	O	00010	2	B

Convert to binary/decimal numbers. Use worksheet 1 to map these numbers to letters.

O	X	O	O	O	8	H	F
X	O	X	O	X	21	U	U
O	O	O	X	X	3	C	Z
O	O	O	X	X	3	C	Z
O	O	O	X	O	2	B	Y

Use the cipher clue to decrypt the letters (be sure to go right-to-left since we are decrypting)

Clue #4 (Vet)

3

Name: _____

Use this index number to select the 3rd name from Clue #1 (after sorting the animals) to get the name FUZZY.

APPENDIX Q

FINAL PROJECT (PET VERSION) RUBRIC

Table Q.1: Rubric for "Pet" Version of the Final Project

	3 Proficient	2 Partially Proficient	1 Unsatisfactory
Clue # 1	Student correctly lists the ten animals in order from lightest to heaviest AND are able to articulate a reasonable process for ordering them.	Student may correctly sort the animals (but may be from heaviest to lightest), but are unable to articulate a reasonable ordering process. OR Student fails to order correctly, but identifies a good process to use.	Student orders animals, but in no discernable pattern and/or Student is unable to articulate a reasonable ordering process.
Numbers & Symbols: Data Representation	Student demonstrates ability to convert between number systems and represent numbers as letters by correctly decoding the message to be "FRIEND."	Student is unable to recognize pattern for binary numbers, but attempts to decode the message by counting the number of '1's. (e.g. BBBBCA)	Student is unable to recognize the pattern for binary numbers and cannot convert numbers to letters or does not attempt the problem.

Table Q.1: Continued

Number & Symbols: Searching	Student gives an answer that can be reasoned to be linear search (i.e., 100) or binary search (7)	Student gives an answer between 1 and 100 that cannot be easily reasoned without knowing their justification.	Student does not attempt the problem or answers a number greater than 100.
Minimal Spanning Tree (Ants)	Student constructs a minimal spanning tree (see answer key) to connect all the anthills AND are able to describe how they arrived at a MST.	Student connects all the anthills in a way that is not a minimal spanning tree, but describes a process to correctly construct a MST OR Student creates a minimal spanning tree, but is unable to describe how they connected the hills.	Student does not connect all the anthills OR Student does not attempt OR Student adds in edges not in the original graph.

Table Q.1: Continued

Finite State Automata (Dog): Diagram	Student uses the six boldface states and attempts to connect them using a discernable flow. To reach proficiency, there needs to exist a fork in the diagram (i.e. a state which leads to two possible states) OR a loop (for potentially infinite schedules). Accuracy compared to the answer key is not important.	Student uses some of the boldface states and attempts to connect them using a discernable flow. There are no forks (i.e., they simply create one possible schedule) or loops, so the diagram resembles a linked list.	Student uses the numbered sentences (each containing multiple states) as their states OR Student does not use any boldface states OR Student does not attempt.
Finite State Automata (Dog): Application and Creation	Student correctly identifies one or more valid schedules (refer to answer key), and is able to come up with a new valid schedule.	Student only completes one of the two pieces: A correct schedule is identified OR A new schedule is provided (that may or may not be correct).	Student identifies the invalid option as the only good schedule OR Student does not attempt creating a new schedule on their own OR Student does not attempt.

Table Q.1: Continued

Optimization (Vet Visit)	Student correctly optimizes the carriers (refer to answer key) and justifies their process.	Student attempts to load the carriers, but arrives at an inefficient solution (but adheres to the 300 lbs constraint). Student still justifies their process.	Student overloads the containers (>300 lbs) OR Student does not attempt.

APPENDIX R

FINAL PROJECT (CARNIVAL VERSION)

Name: _____

Carnytown Carnival Murder Mystery

While visiting the Carnytown Carnival, you found a dead body! Can you help solve the murder at the carnival before it's too late and the carnival gets shut down forever?

Collect clues from different carnival members by completing their challenge. Then, pair the clues together to see if you can uncover a secret message.

Clue #1

(This is your free clue – Complete other challenges to get more)
Each person has their age next to their name.


Tammy, 19


Larry, 21


Garry, 27


Gerri, 29


Perry, 33


Harry, 42


Barry, 37


Terry, 25


Sammy, 23


Kerry, 40

It might be useful to write down the lineup in order from youngest to oldest:

_____

What process did you use to put them in order?

_____

Name: _____

Detective Notes: Sammy

Sammy remembers Odin's obsession with the number "2." Sammy wrote down the following two problems, which he recalls as some of Odin's favorites. (If you get stuck, think about a base-2 number system).

Odin's First Challenge

Odin wrote messages using numbers. The number system Odin used only had **TWO** symbols: a horizontal line (-) and a vertical line (|).

Can you decode the last message from Odin?

-	|	|	-	|	
-	-	-	-	|	
-	-	|	|	|	
-	|	-	-	|	
-	-	-	|	|	

Odin's Second Challenge

Odin's library had over 1,000 different books. Odin kept everything alphabetized.

If Odin wanted to read a book, he had to search through his library to find it (he knew the title of the book). How many items did Odin have to look at in order to find what he was looking for (or decide it was not in his library collection)?

Write your answer here: _____

Here are the first few letters from Odin's system.

A = - - - - |
B = - - - | -
C = - - - | |
D = - - | - -

Below, we've listed each letter and its numerical place in the alphabet.

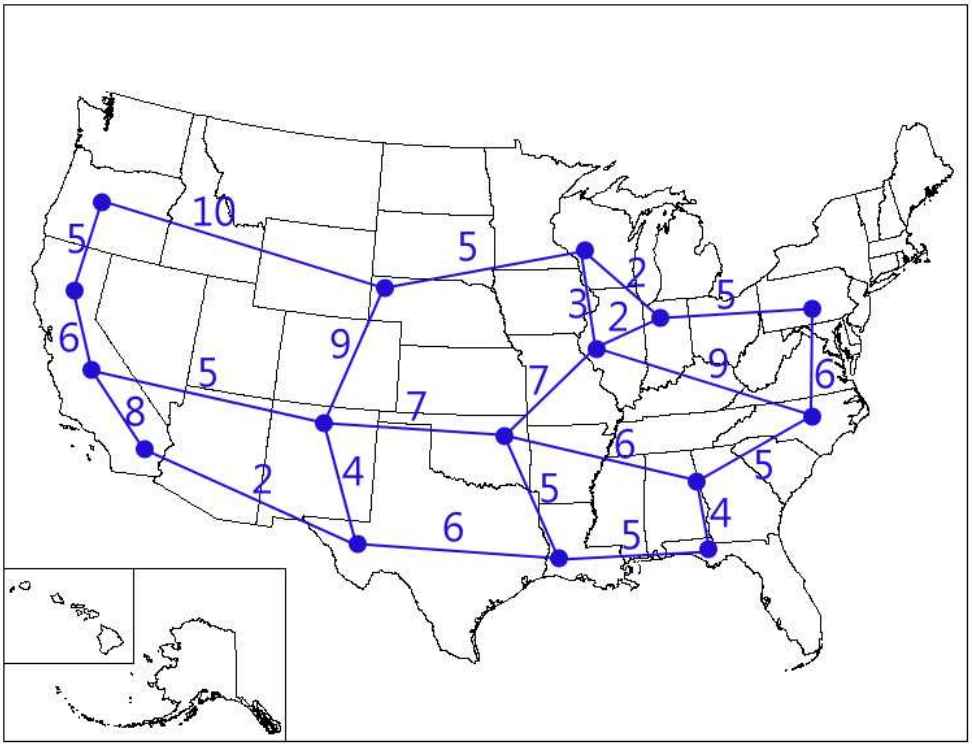
A = 1
B = 2
C = 3
D = 4
E = 5
F = 6
G = 7
H = 8
I = 9
J = 10
K = 11
L = 12
M = 13
N = 14
O = 15
P = 16
Q = 17
R = 18
S = 19
T = 20
U = 21
V = 22
W = 23
X = 24
Y = 25
Z = 26

Name: _____

Detective Notes: Tammy

Tammy helps send all the carnival equipment on trains to the next city where the carnival will set up. In the map below, each dot is a city where the carnival will be this year. The numbers next to each line represent the cost (in millions) to build railroad tracks between those two cities. Tammy needs your help in connecting all of the cities using the smallest amount of cash.

Shade the lines where the railroad tracks should be laid so that the railroad company uses the least amount of money to build the tracks. Once complete, you will be given your next clue.



How did you decide which routes to build railroad tracks?

_____

_____

_____

Name: _____

Detective Notes: Larry

Larry will give you a clue if you help him plan the carnival festival.

Given the following rules, can any of the events can be scheduled without conflicting? Only one event can ever be in progress at a time because the carnival only has one stage.

The Rules (the events are in bold):

1. The **"Roffles"** will play as long as the **"Mezzos"** play after them.
2. **Speeches** must happen before **fireworks**, and before the **"Mezzos"** play.
3. **"Mezzos"** plays before the **party** starts.
4. The **choir** will sing before the "**Roffles,**" or before the **speeches** are given.
5. The "**Roffles**" must play before the **speeches** are given.
6. The **party** ends before the **lottery** begins.
7. The carnival will **thank its sponsors** after the **fireworks** or after the **lottery**.
8. The **fireworks** start after the **lottery**.

Use the blank space below to work on a solution to this problem by organizing the events and transitions.

Circle any of the following schedules if they follow the carnivals's rules. Additionally, **use the bottom of the page to write another schedule that would work.**

1. Choir
2. Speeches
3. Fireworks
4. Thank Sponsors

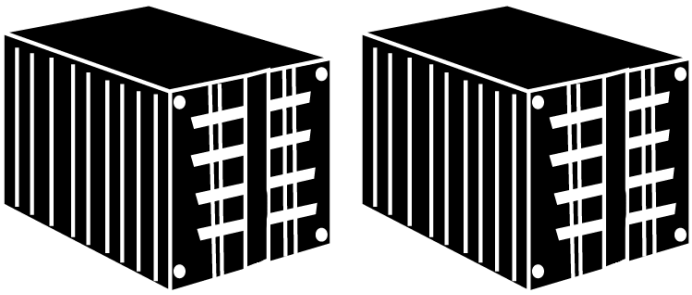
1. Choir
2. Roffles
3. Start the Party
4. Speeches

1. Start the Party
2. Lottery
3. Fireworks
4. Thank Sponsors

Name: _____

Detective Notes: Terry

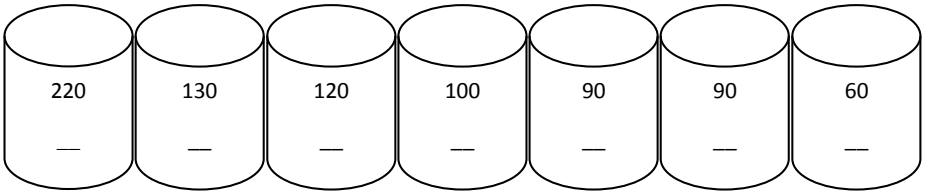
The only bit of information Terry knows is a number Odin always whispered to her. She'll tell you the number if you help her out before a storm hits the carnival.



Container 1 Container 2

A storm is coming, and Terry needs to move some of the rides and equipment inside big storage containers. She wants to maximize the total amount of weight in both containers. Each container can only hold 300 tons, or else the wheels will burst.

Which weights below should Terry put into the containers? **Beneath the numbers inside each weight, write the number of the container it should go into. She won't be able to fit everything, so some weights will not have a container number.**



How did you choose the *first* piece of machinery to put into the storage container?

_____

_____

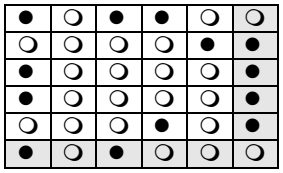
Name: _____

Larry's Clue

A	→	A
B	→	D
C	→	F
D	→	G
E	→	E
F	→	H
G	→	J
H	→	K
I	→	I
J	→	L
K	→	M
L	→	N
M	→	P
N	→	Q
O	→	O
P	→	R
Q	→	S
R	→	T
S	→	V
T	→	W
U	→	U
V	→	X
W	→	Y
X	→	Z
Y	→	B
Z	→	C

Tammy's Clue

Two symbols are used in the following grid. Tammy included a parity row/column in case someone messed up the message before you received it.



Terry's Clue

3

APPENDIX S

FINAL PROJECT (CARNIVAL VERSION) ANSWER KEY

Name: _____

Carnytown Carnival Murder Mystery

While visiting the Carnytown Carnival, you found a dead body! Can you help solve the murder at the carnival before it's too late and the carnival gets shut down forever?

Collect clues from different carnival members by completing their challenge. Then, pair the clues together to see if you can uncover a secret message.

Clue #1

(This is your free clue – Complete other challenges to get more)
Each person has their age next to their name.


Tammy, 19


Larry, 21


Garry, 27


Gerri, 29


Perry, 33


Harry, 42


Barry, 37


Terry, 25


Sammy, 23


Kerry, 40

Using their ages to sort them youngest to oldest:

Tammy, Larry, Sammy, Terry, Garry, Gerri, Perry, Barry, Kerry, Harry

Use one of the clues (the number 3) to choose the third index (assuming 1-indexing): Sammy.

Name: _____

Detective Notes: Sammy

Sammy remembers Odin's obsession with the number "2." Sammy wrote down the following two problems, which he recalls as some of Odin's favorites. (If you get stuck, think about a base-2 number system).

Odin wrote messages using numbers. The number system Odin used only had **TWO** symbols: a horizontal line (-) and a vertical line (|).

Can you decode the last message from Odin?

-	|	|	-	|	**01101**	**13**	**M**
-	-	-	-	|	**00001**	**1**	**A**
-	-	|	|	|	**00111**	**7**	**G**
-	|	-	-	|	**01001**	**9**	**I**
-	-	-	|	|	**00011**	**3**	**C**

Odin's library had over 1,000 different books. Odin kept everything alphabetized.

If Odin wanted to read a book, he had to search through his library to find it (he knew the title of the book). How many items did Odin have to look at in order to find what he was looking for (or decide it was not in his library collection)?

Write your answer here: _____

Log_2(1000) = 10 (you would need to look at 10 books in a worst case scenario to determine if the book is in your collection).

Here are the first few letters from Odin's system.

A = - - - - |
B = - - - | -
C = - - - | |
D = - - | - -

Below, we've listed each letter and its numerical place in the alphabet.

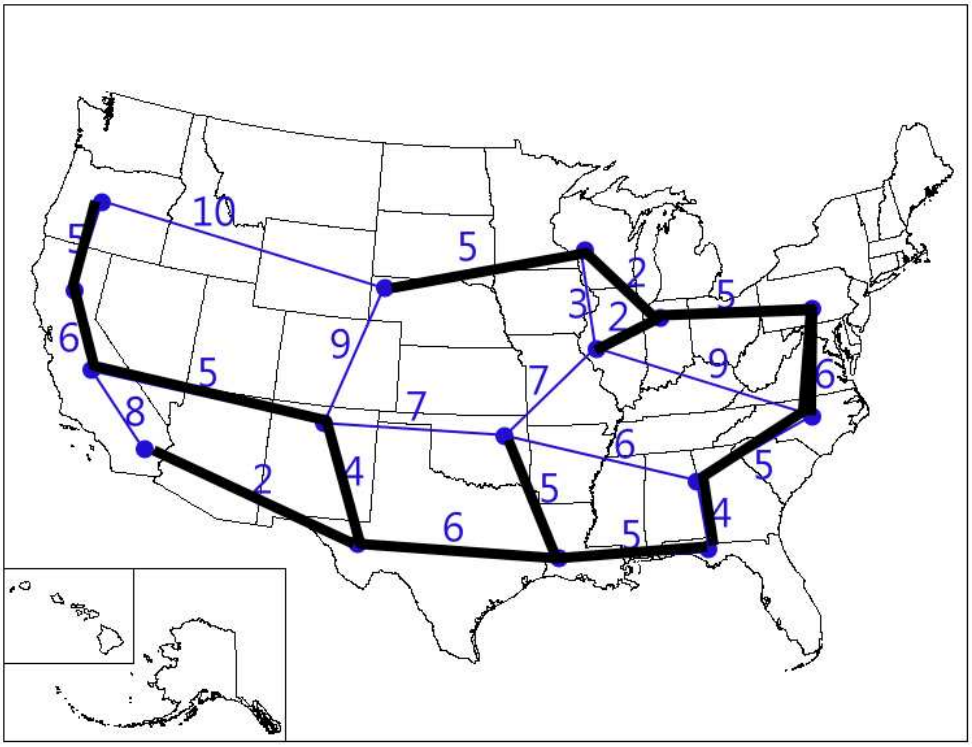
A = 1
B = 2
C = 3
D = 4
E = 5
F = 6
G = 7
H = 8
I = 9
J = 10
K = 11
L = 12
M = 13
N = 14
O = 15
P = 16
Q = 17
R = 18
S = 19
T = 20
U = 21
V = 22
W = 23
X = 24
Y = 25
Z = 26

Name: _____

Detective Notes: Tammy

Tammy helps send all the carnival equipment on trains to the next city where the carnival will set up. In the map below, each dot is a city where the carnival will be this year. The numbers next to each line represent the cost (in millions) to build railroad tracks between those two cities. Tammy needs your help in connecting all of the cities using the smallest amount of cash.

Shade the lines where the railroad tracks should be laid so that the railroad company uses the least amount of money to build the tracks. Once complete, you will be given your next clue.



How did you decide which routes to build railroad tracks?

Using Kruskal's algorithm, you can construct a minimal spanning tree without doing any addition.

Name: _____

Detective Notes: Larry

Larry will give you a clue if you help him plan the carnival festival.

Given the following rules, can any of the events can be scheduled without conflicting? Only one event can ever be in progress at a time because the carnival only has one stage.

The Rules (the events are in bold):

1. The **"Roffles"** will play as long as the **"Mezzos"** play after them.
2. **Speeches** must happen before **fireworks**, and before the **"Mezzos"** play.
3. **"Mezzos"** plays before the **party** starts.
4. The **choir** will sing before the "**Roffles,**" or before the **speeches** are given.
5. The "**Roffles**" must play before the **speeches** are given.
6. The **party** ends before the **lottery** begins.
7. The carnival will **thank its sponsors** after the **fireworks** or after the **lottery**.
8. The **fireworks** start after the **lottery**.

Use the blank space below to work on a solution to this problem by organizing the events and transitions.

The first column below is the only correct ordering of events.

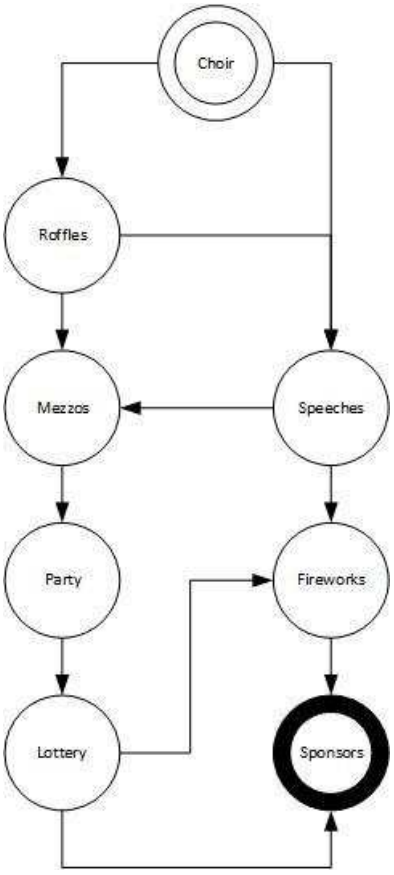
For scoring purposes, let's count both the first and third columns as correct (the third column is a valid ordering of events, but it does not start with the choir, which was marked as a confusing aspect during deployment).

Circle any of the following schedules if they follow the carnivals's rules. Additionally, **use the bottom of the page to write another schedule that would work.**

1. Choir	1. Choir	1. Start the Party
2. Speeches	2. Roffles	2. Lottery
3. Fireworks	3. Start the Party	3. Fireworks
4. Thank Sponsors	4. Speeches	4. Thank Sponsors

[Answers may vary]

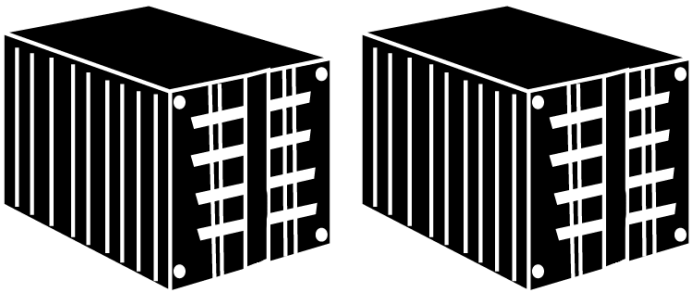
Name: _____


Choir
Roffles
Mezzos
Speeches
Party
Fireworks
Lottery
Sponsors

Name: _____

Detective Notes: Terry

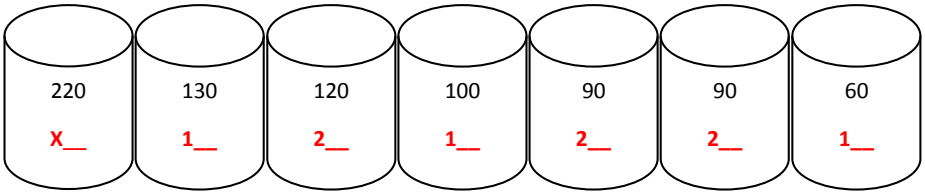
The only bit of information Terry knows is a number Odin always whispered to her. She'll tell you the number if you help her out before a storm hits the carnival.



Container 1 Container 2

A storm is coming, and Terry needs to move some of the rides and equipment inside big storage containers. She wants to maximize the total amount of weight in both containers. Each container can only hold 300 tons, or else the wheels will burst.

Which weights below should Terry put into the containers? **Beneath the numbers inside each weight, write the number of the container it should go into. She won't be able to fit everything, so some weights will not have a container number.**



How did you choose the *first* piece of machinery to put into the storage container?

_____

_____

Name: _____

Larry's Clue

A	→	A
B	→	D
C	→	F
D	→	G
E	→	E
F	→	H
G	→	J
H	→	K
I	→	I
J	→	L
K	→	M
L	→	N
M	→	P
N	→	Q
O	→	O
P	→	R
Q	→	S
R	→	T
S	→	V
T	→	W
U	→	U
V	→	X
W	→	Y
X	→	Z
Y	→	B
Z	→	C

Tammy's Clue

Two symbols are used in the following grid. Tammy included a parity row/column in case someone messed up the message before you received it.

●	❍	●	●	❍	❍
❍	❍	❍	❍	●	●
●	❍	❍	❍	❍	●
●	❍	❍	❍	❍	●
❍	❍	❍	●	❍	●
●	❍	●	❍	❍	❍

Note that this looks like binary and error detection.

●	❍	●	●	●	❍
❍	❍	❍	❍	●	●
●	❍	❍	❍	❍	●
●	❍	❍	❍	❍	●
❍	❍	❍	●	❍	●
●	❍	●	❍	❍	❍

Use parity bits to correct the error.

●	❍	●	●	●	10111	23	W
❍	❍	❍	❍	●	00001	1	A
●	❍	❍	❍	❍	10000	16	P
●	❍	❍	❍	❍	10000	16	P
❍	❍	❍	●	❍	00010	2	B

Convert to binary/decimal, then use number/letter mapping to get letters.

●	❍	●	●	❍	W	T
❍	❍	❍	❍	●	A	A
●	❍	❍	❍	❍	P	M
●	❍	❍	❍	❍	P	M
❍	❍	❍	●	❍	B	Y

Use the cipher clue to decrypt message.

Terry's Clue

Name: _____

3

APPENDIX T

FINAL PROJECT (CARNIVAL VERSION) RUBRIC

Table T.1: Rubric for the "Carnival" Version of the Final Project

	3 Proficient	2 Partially Proficient	1 Unsatisfactory
Clue # 1	Student correctly lists the ten people in order from lightest to heaviest AND are able to articulate a reasonable process for ordering them.	Student may correctly sort the people (but may be from heaviest to lightest), but are unable to articulate a reasonable ordering process. OR Student fails to order correctly, but identifies a good process to use.	Student orders people, but in no discernable pattern and/or Student is unable to articulate a reasonable ordering process.
Numbers & Symbols: Data Representation	Student demonstrates ability to convert between number systems and represent numbers as letters by correctly decoding the message to be "MAGIC."	Student is unable to recognize pattern for binary numbers, but attempts to decode the message by counting the number of '1's. (e.g. CACBB)	Student is unable to recognize the pattern for binary numbers and cannot convert numbers to letters or does not attempt the problem.

Table T.1: Continued

Number & Symbols: Searching	Student gives an answer that can be reasoned to be linear search (i.e., 1000) or binary search (10)	Student gives an answer between 1 and 1000 that cannot be easily reasoned without knowing their justification.	Student does not attempt the problem or answers a number greater than 1000.
Minimal Spanning Tree (Planes)	Student constructs a minimal spanning tree (see answer key) to connect all the cities AND are able to describe how they arrived at a MST.	Student connects all the cities in a way that is not a minimal spanning tree, but describes a process to correctly construct a MST OR Student creates a minimal spanning tree, but is unable to describe how they connected the cities.	Student does not connect all the cities OR Student does not attempt OR Student adds in edges not in the original graph.

Table T.1: Continued

Finite State Automata (Carnival Event): Diagram	Student uses the eight boldface states and attempts to connect them using a discernable flow. To reach proficiency, there needs to exist a fork in the diagram (i.e. a state which leads to two possible states) OR a loop (for potentially infinite schedules). Accuracy compared to the answer key is not important.	Student uses some of the boldface states and attempts to connect them using a discernable flow. There are no forks (i.e., they simply create one possible schedule) or loops, so the diagram resembles a linked list.	Student uses the numbered sentences (each containing multiple states) as their states OR Student does not use any boldface states OR Student does not attempt.
Finite State Automata (Carnival Event): Application and Creation	Student correctly identifies one or more valid schedules (refer to answer key), and is able to come up with a new valid schedule. For scoring purposes, both the first and third columns will be accepted as correct.	Student only completes one of the two pieces: A correct schedule is identified OR A new schedule is provided (that may or may not be correct).	Student identifies the invalid option as the only good schedule OR Student does not attempt creating a new schedule on their own OR Student does not attempt.

Table T.1: Continued

Optimization (Machinery Storage)	Student correctly optimizes the carriers (refer to answer key) and justifies their process.	Student attempts to load the carriers, but arrives at an inefficient solution (but adheres to the 300 ton constraint). Student still justifies their process.	Student overloads the containers (>300 tons) OR Student does not attempt.