

Hamline University

DigitalCommons@Hamline

School of Education Student Capstone Projects

School of Education

Fall 2019

How can second-grade students learn algorithmic thinking and pattern recognition through collaborative learning?

Christine Walth

Follow this and additional works at: https://digitalcommons.hamline.edu/hse_cp



Part of the Education Commons

HOW CAN SECOND-GRADERS LEARN ALGORITHMIC THINKING AND
PATTERN RECOGNITION THROUGH COLLABORATIVE LEARNING?

By

Christine Walth

A capstone submitted in partial fulfillment of the requirements
for the degree of Master of Arts in Teaching

Hamline University

Saint Paul, Minnesota

December 2019

Capstone Project Facilitator: Jennifer Carlson and Feride Erku

Content Expert: Kristine Wobbema

Peer Reviewer: Gary K. Walth

ACKNOWLEDGEMENTS

This capstone would not be possible without the assistance of Hamline professors Jennifer Carlson and Feride Erku. Their attention to detail and support of students through this work should not go unnoticed. I would also like to thank Kristine Wobbema, my content expert, whom I admire as a mentor for her organization, technological expertise, and ability to share the love of learning with each student in which she interacts. Finally, I would like to express gratitude to my family who have supported me through graduate school and all my teaching endeavors. The work on this project is dedicated to the memory of my great aunt Therese, a fellow educator, who encouraged me to pursue my Master's degree and helped me to get started. Aunt Therese's character and her influence in the lives of others is unparalleled, and I hope to follow her legacy throughout my educational career.

Abstract

In today's technology-filled world, employers seek applicants with strong computational thinking (CT) skills and computer science (CS) backgrounds. The demand for CT education reaches all the way to the elementary level. Wing (2006) states "computational thinking is a fundamental skill for everyone, not just for computer scientists" (p. 33). Though researchers are continuing to define all aspects of CT, the major elements include: algorithmic thinking, pattern recognition, decomposition, and abstraction. The digital age has also caused an increase in screen time, time children spend in front of a device, which has prompted studies on the negative physical and psychological effects it can have on children. Scoggin (2018) explains that school students are demonstrating a lack of social skills due to increased screen time in the classroom. As a response to this research, this capstone builds on relevant studies and provides a unit of lessons to answer the question: *How can second-grade students learn algorithmic thinking and pattern recognition through collaborative learning?* The detailed project includes cooperative activities and assessments to teach CT skills without the use of devices.

TABLE OF CONTENTS

CHAPTER ONE: Introduction	5
Personal Background	6
Professional Significance	7
Guiding Questions	9
Conclusion	10
CHAPTER TWO: A Review of the Literature	12
Computational Thinking	12
Aspects of Computational Thinking	13
Computational Thinking in Education	16
Screen Time	20
Physical Effects of Screen Time	20
Psychological Effects of Screen Time	22
Screen Time in Schools	23
Cooperative Learning	24
Successful Implementation of Cooperative Learning	25
Learning Computational Thinking “Unplugged”	26
CHAPTER THREE: Project	29

Curriculum Design	29
Project Framework	30
Setting and Students	33
Background of Staff	34
CHAPTER FOUR: Conclusion	36
Summary of Literature Review	36
Description of the Project	37
Limitations of the Project	38
Personal Reflection	39
REFERENCES	42
APPENDIX A: Lesson Plans	49
Lesson One: Introduction to Algorithms	51
Lesson Two: Constructing Algorithms	55
Lesson Three: Following Algorithms	61
Lesson Four: Revising Algorithms	68
Lesson Five: Patterns in Algorithms	75
Lesson Six: Composing a Detailed Algorithm	82

CHAPTER ONE

Introduction

Overview

Elementary schools today have taken a turn towards a largely technology-filled learning environment. Each content area has apps and programs available to help young people be exposed to information and practice. Classrooms are furnished with tablets, laptops, and computers, often in a 1:1 ratio of student to device. The drive for technology is a reflection of our increasingly computer-centric world.

The number of computer science jobs continues to grow at a much faster rate than that of most occupations. Jobs in computer science are expected to grow by 19% over the next ten years (Bureau of Labor Statistics, U.S. Department of Labor, 2016). Schools are therefore introducing computer science (CS) into their curricula not only to encourage students to think about CS as a future career but to give students skills that they will need to fit into a technology-filled market. A large part of the CS curriculum, especially at the elementary level, is learning computational thinking (CT).

Over the last ten years, the concept of “screen time” has become a hot-button issue (Scoggin & Vander Ark, 2018). This capstone will talk in detail about the concern around screen time: amount children spend in front of a device (television, computer, tablet, etc.) both at home and in schools (Lissak, 2018).

The devices in students’ hands, the push for teaching computational thinking skills, and the concern for too much screen time have left me wondering: *How can second-grade students learn algorithmic thinking and pattern recognition through collaborative learning?*

In Chapter One, I will explain the background of my desire to study the subject and how I arrived on my research question. The chapter will start with my personal background, move on to my professional experience, and finally provide an overview of the capstone project.

Personal Background

Since I was a child, I have always been fascinated with technology. After school, I would often rush home to get on the large tabletop iMac that my family shared. I was interested in playing games, reading the latest celebrity news, and chatting with my friends through instant messenger. I increased my knowledge of the computer with trial and error and was soon the one my parents and sister would go to when the computer was not working properly.

Throughout high school and college, I continued to enjoy time on my devices. It became fascinating to learn how to work with the computer’s system to override certain

boundaries, enhance the computer's graphics and software, and download all kinds of content.

When I began student teaching in 2014, I noticed how beneficial it was to know how to work with computers. Helping my cooperating teacher figure out how to navigate new programs or access helpful websites for teaching was a strength of mine. In interviews for positions, I was always confident in my ability to talk about my appreciation and knowledge of technology.

In 2016, I was hired in a permanent position teaching second grade in a large suburban district in the Midwest. Though most of my team was confident with technology, soon my role as the problem-solver became clear when it came to anything with computers or devices. From online grading to making new slideshows to navigating and sharing information about new apps on the iPads, my team looked to me to lead that area of our teaching. I joined the technology team at our school and assisted in implementing the Hour of Code (an initiative that teaches students how to begin coding in K-6 classrooms). In my second year, my classroom was selected to receive 1:1 iPads, compared to the 1:3 ratio used by most classrooms.

Professional Significance

As a supporter of technology in education and interested in the best intentions for our students, I was asked last year (2019) to be a part of the district's selected committee to decide what devices K-2 students will be using in the 2020-21 school year. A major influence in our decision was the ratio of students to devices. At first, my hope was to

get as close to a 1:1 ratio as possible for each K-2 classroom. However, my opinion soon began to change as I listened to other teachers on the committee.

Several of them were concerned about screen time that students were receiving in school. They argued that students were losing opportunities to communicate with one another and time to practice social skills was replaced by time on devices. Specifically, one of the experienced teachers on the committee indicated that she noticed a difference in students since computerized learning became so prevalent during the learning day. She emphasized the importance of social practice in the early elementary years on a child's development.

This was the first time that I questioned the amount of screen time students were exposed to in the classroom. The effects of screen time on young children was something I began to research. It was especially interesting to me because I am a strong advocate for new technology and apps that are built for elementary students. My preliminary investigation suggested that too much screen time may have a negative impact (Gingold, Simon, & Schoendorf, 2014).

At the end of the 2018-19 school year, my principal invited me to go to the MN Code Summit, to learn more about coding, technology, and computer education. It was at this time that I learned that my school was becoming a "Computer Science Immersion" school in the district. I was immediately filled with excitement about this move for our school. While attending professional development sessions on computer science education and computational thinking, I continued to think back to the concerns of the device committee.

When the decision of becoming a “Computer Science Immersion” school was announced to my staff, a few staff members raised their concerns. One kindergarten teacher said she was worried that parents would worry about the change. As a mother, when she hears “computer science,” she automatically pictures children on devices and worries about the impact of more screen time in schools, thus was concerned that other parents would feel the same way and wondered how we could calm their fears. A district representative assured us that computer science does not necessarily mean more screen time and that there are opportunities to teach computational thinking without devices. Other research, including Paul (2015) and Sung (2019), have made similar arguments that it is possible to teach computational thinking skills without devices or “unplugged.”

Guiding Questions

These experiences brought me to my main research question: *How can K-2 teachers teach computational thinking skills without devices?* Before I began to answer this question and conduct research, I had to look carefully at other issues to make the purpose for research more clear: the effects of screen time on children, the push for computer science education, the relationship between computer science and computational thinking and what elements of computational thinking should be taught to K-2 students. Each of these topics will be explored in Chapter Two.

I was able to find a large number of studies on screen time both at home and in schools, such as Domingues (2017) and Lissak (2018). Articles about computer science education and computational thinking were also numerous and seem to be growing by the number each year, Denning (2009) and Selby and Woollard (2013) are two such

examples of articles about computational thinking. Examples of research regarding computer science education include Yadav, Hong, and Stephenson (2016) and Tran (2019.) It was more difficult to find resources that specifically target how to teach elements of computational thinking without devices. Peel, Sadler, and Friedrichsen (2019) implemented unplugged lessons for middle and high school.

There is numerous research on collaborative learning, including Erdogan (2019) and Slavin (2015); however, I could not locate any sources that have attempted to study teaching CT through collaborative learning. Therefore, I believe my research will be helpful to fill the gap in research to answer: *How can second-grade students learn algorithmic thinking and pattern recognition through collaborative learning?* If K-2 teachers are looking for ways to teach computational thinking as a part of computer science education, but are concerned about increasing screen time, they can use this research to teach these skills without students on devices.

Conclusion

Chapter One has provided my background and interest in computational thinking, the significant impact that this question has on my job as a second-grade teacher, and the need for further research in the area of teaching without devices in order to decrease screen time in schools. It is my goal to write a unit of six lessons for second graders to teach algorithmic thinking and pattern recognition. These lessons will use collaborative learning and will not use student devices. In this way, I can contribute to understanding the research question: *How can second-grade students learn algorithmic thinking and pattern recognition through collaborative learning?*

Overview of Remaining Chapters

In Chapter Two, I introduce and summarize relevant research on my topic. I will start with a look at the effects of screen time on young children; then explain the elements of computer science and computational thinking and education's recent push to introduce these into K-12 classrooms. Finally, I look in-depth at why it is important that K-2 teachers be able to teach computational thinking skills without increasing screen time in schools.

In Chapter Three, I describe my plan to implement a device-free curriculum to teach K-2 computational thinking skills, which will include an outline of each lesson, the assessments, standards, and the learning targets for each lesson.

CHAPTER TWO

Literature Review

Overview

This literature review was conducted in order to develop an understanding of the history, recent studies, and projects that have been conducted on the relevant topics for the research question being investigated in this capstone: *How can second-grade students learn algorithmic thinking and pattern recognition through collaborative learning?*

The first subject explored in this chapter will be computational thinking, its importance in the world, modern education, and elementary programs. The next section will review research on the effects of screen time on physical and psychological effects and the limited research on screen time in schools. The third subject covers collaborative learning, a method of teaching that involves social interaction in small group settings. Finally, the review will cover research relating directly to learning computational thinking without screens and will point to a gap in research that this capstone will address.

Computational Thinking

Computational Thinking (CT) is a system of understanding and solving problems, whether used by humans or computers. It involves the ability to logically approach solutions and concentrates on specific processes of thinking. Wing (2006) provides one of the well-known definitions of CT describing it as a combination of “solving problems, designing systems, and understanding human behavior, by drawing on the concepts fundamental to computer science” (p. 33).

This section begins by looking at how research has evolved around CT and its relation to computer science. Additional research focuses on algorithmic thinking and pattern recognition, which will be further explored in Chapter Three. Next, this section will focus on the recent push for CT skills in K-12 education as action research reports have been done in the last few years on CT being implemented in classrooms around the world. Finally, this chapter will look at studies of CT being taught unplugged or without devices, which is the specific focus of the current capstone. It is important to focus on the elements of CT in order to understand the rationale behind bringing these skills into the primary grades.

Aspects of Computational Thinking

Computer science is defined as “the study of the design and operation of computer hardware and software, and of the application of computer technology to science, business, and the arts” (*The American Heritage® Science Dictionary*, 2011). Though CT is a required component of computer science, it covers a broader scope of knowledge and skills.

Computer Science (CS) related jobs have become one of the largest growing career fields in the world. In a recent study Krieger (2019) states, “undergraduate computer science enrollments in research universities in the United States and Canada tripled between 2006 and 2016” (p. 1). Education is quickly adapting to meet the needs of an increasingly computer-centric world.

Wing (2006) is often credited for giving a modern definition to the term “computational thinking.” Not only does she describe it as a means to evaluate problems and seek out solutions by combining the advantages of the human mind and the skills of a computer, but she opened the door for CT to all. “Computational thinking is a fundamental skill for everyone, not just for computer scientists” (Wing, 2006, p. 33). Wing goes on to explain what CT is and what it is not, re-emphasizing that CT is used in many different fields and can be accessed by all. The areas of CT that she specifically points out include: abstraction, mathematical and engineering thinking, looking for patterns, and developing algorithms (Wing, 2006).

Selby and Woollard (2013) expanded Wing’s definition to include other aspects of CT. In an attempt to create a more complete definition, Selby and Woollard (2013) included the following elements:

- A Thought Process. Computational Thinking is first and foremost a process of thinking that breaks apart problems in an effort to find solutions. This often includes developing algorithmic expressions that can often be used by an information-processing agent.

- Abstraction. The process of abstraction involves generalizing a problem and removing information that is not necessary in order to focus on a clear idea of the problem.
- Decomposition. Breaking down large problems into small, specific steps is another essential part of computational thinking.

Selby and Woollard evaluate a list of terms in the search for an accurate definition of CT. Besides the three terms mentioned, they also recommend adding algorithmic thinking, generalization, and evaluation (Selby & Woollard, 2013).

Kalelioğlu, Kukul, & Gülbahar (2016) also undertook the task of defining CT under one umbrella, and after looking over an extensive amount of research about CT, these authors inputted terms used in various reports into a Wordle:

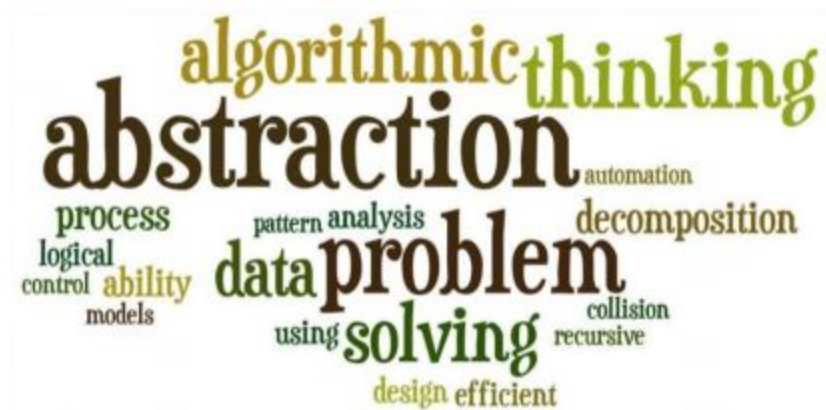


Figure 1. Most commonly used words in the definitions of CT. Reprinted from “A Framework for Computational Thinking Based on a Systematic Research Review” by Kalelioğlu, F., Kukul, V., & Gülbahar, Y. (2016), *Baltic J. Modern Computing*, 4(3), p. 584.

Many of the same terms brought up in Wing (2006) and Selby and Woollard (2013) appear in Figure 1, providing a similar definition for CT as seen in other research. Kalelioğlu et al. (2016) go on to evaluate papers written about CT between 2006-2014. More than half of the research investigated discussed CT education. Kalelioğlu, et al. (2016) note the importance of CT understanding for educators as the need for pedagogy to teach CT expands. Thus, the next section will examine CT in education. It will inspect the elements that should be taught at the primary, K-2, level and the importance of CT in current school systems.

Computational Thinking in Education

“Computer science and the technologies it enables rest at the heart of our economy and the way we live our lives. To be well-educated citizens in a computing-intensive world and to be prepared for careers in the 21st century, our students must have a clear understanding of the principles and practices of computer science.” (CSTA, 2011).

The Computer Science Teachers Association (CSTA) was created in 2004 by the Association for Computing Machinery as an effort to enhance computer science education in K-12. CSTA has released standards that many schools are using to build their computer science education programs. Many of these standards relate directly to CT. A few examples of these standards in the K-2 category include:

- Identify and describe patterns in data visualizations, such as charts or graphs, to make predictions.

- Decompose (break down) the steps needed to solve a problem into a precise sequence of instructions.
- Debug (identify and fix) errors in an algorithm or program that includes sequences and simple loops (CSTA, 2017, p. 4-5).

Challenges, methods, and benefits of implementing K-12 CT education have been researched extensively. In some of this research, the phrases computer science, computation, and computational thinking have been used interchangeably.

Lee, Martin, Denner, Coulter, Allan, Erickson, Werner (2011) delve into what computational thinking (CT) education for K-12 might look like in practice. The researchers developed three activities for high school students: Modeling and Simulation, Robotics, and Game Design & Development. To further illustrate the focus on CT, they were able to find how each activity promoted abstraction, automation, and analysis. The activities motivated the students and expanded their knowledge of CT. Lee et al. (2011) concluded “that future [research] efforts get more specific about the type and level of CT that will be addressed” (p. 36).

There has been further research done investigating CT in the elementary school which applies more directly to the topic of the current capstone. Yadav, Hong, and Stephenson, (2016) and Angeli, Voogt, Fluck, Webb, Cox, Malyn-Smith, and Zagami (2016) discuss the importance of elementary CT education. Yadav et al. emphasize that CT can be simplified enough for elementary students to understand, “for example, elementary students can learn about algorithms by breaking down a simple daily task,

such as brushing teeth, into a sequence of steps” (p. 566). The article suggests tools that teachers can use, such as the CSTA standards.

Angeli et al. (2016) while encouraging elementary CT education, inspect some of the challenges that implementing a CT program could have on educators. One challenge is a lack of available curriculum in CT for elementary students. The elements of CT are broken into five distinct areas: abstraction, generalization, decomposition, algorithms, and debugging. Angeli et al. developed their own framework of how each of the areas might look in a curriculum for K-2, 3-4, and 5-6. In relation to algorithms, one of the main focuses of the capstone project, Angeli et al. (2016) created the target for K-2 students to: “define a series of steps for a solution” and “put instructions in the correct order” (p. 51). Another challenge that is examined in the article is the prior knowledge that teachers must have in order to accurately teach CT. The researchers suggested professional development that implements TPACK, Technological Pedagogical Content Knowledge, for a well-rounded approach for teachers to become familiar with teaching CT material. (Angeli et al, 2016).

The need for professional development for teachers to competently teach CT is echoed by Ozturk, Dooley, and Welch (2018). In this action research, data was gathered within the first six months of an elementary school that was embedding CT into its curriculum. This data included teacher interviews, field notes from training, and artifacts from teaching CT in the classroom. While the benefits of teaching included an increase in student collaboration and giving students more autonomy and ownership of their work,

there were also concerns about having enough time to plan and execute the lessons. (Ozturk, Dooley, & Welch, 2018).

There has been considerable research done on CT in elementary schools. One example is from Price and Price-Mohr (2018) who conducted research for a group of 7-11 year-old children in England. In this study, the children worked with a Java-based application to code or write out the directions for a character and adjust the background or setting. With nearly no prior knowledge of CT, most children were motivated to figure out steps in the program to display elements of decomposition, abstraction, and logical thinking by animating stories within the application. In the conclusion, Price and Price-Mohr “encourage [educators] to critically evaluate concepts within CT and to teach [those] linked closely to programming” (p. 299).

Another study was conducted by Tran (2019) with third-grade students using CT skills. Lessons were conducted once a week with online applications to give students experience coding. Tran noted that the students who coded had grown more in mathematics and problem solving from those who had not coded. Students showed an increase in skill from a pre-test to a post-test in the areas of sequence, algorithms, loops, debugging, and conditional coding.

The research in CT presented in this chapter has been about learning CT skills with computers and devices. The next area of this literature review will focus on information about screen time. The combination of topics will lead to the purpose of this research project and capstone which focuses on teaching computational thinking without increasing screen time.

Screen Time

Screen time is measured as the amount of time daily spent in front of a digital device such as a television, computer, or tablet. As the digital age continues to grow inside the classroom and out, researchers question the effects of screen time on youth and adults. Studies, such as Domingues' (2017), have looked at the negative physiological and psychological effects children face when exposed to too much screen time on a daily basis. Parents and teachers are concerned about the adverse effects screen time has on brain function in comparison to the benefits of brain function that reading creates (Horowitz-Kraus, 2017). This section will discuss the effects of screen time in children, including screen time within the school day.

Physical Effects of Screen Time

In recent years, parents, teachers, and healthcare professionals have begun to worry about the adverse effects of screen time on physical health. The following studies, among others, reveal how screen time can change children's overall body wellness.

A study from Foltz, Cook, Szilagyi, Auinger, Stewart, Bucher, and Baldwin (2011) examines the physical effects on children based on their screen time usage. The aim of this study was to compare obesity rates with factors that influence them: nutrition, sugar, physical activity, and screen time. Foltz et al (2011) states in order to decrease the national average, "1 in 6 US adolescents are obese," (p. 424), Blue Cross Blue Shield of Massachusetts and the Maine Youth Overweight Collaborative developed a campaign for families to follow for children; 5-2-1-0 recommends "5 or more servings of fruits and vegetables, <2 hours of screen time (TV, computer, video games), at least 1 hour of

physical activity, and no (0) sugar-sweetened beverages” (p. 424). The results of this report indicate that children who were exposed to more than two hours of screen time were more likely to be obese. One-fourth of the children surveyed were using more than five hours of screen time a day.

The results are echoed in articles by Gingold, Simon, & Schoendorf (2014) and Domingues (2017). Gingold et al. also recommends less than two hours of screen time a day, but found that almost half of 6-17 year olds did not meet these requirements. They note that physical activity is directly effected by screen time. The combination of a lack of physical activity and increased screen time shows a correlation with increased BMI. Domingues (2017) aims to study the negative effects of screen time, even when recommendations for physical activity are being met. The report examines statistics of children with increased screen time and the effects on their health including: lower physical strength, general physical complaints, in particular, headache and backache, and interrupted or shorter sleep duration (Domingues, 2017).

Lissak (2018) also looks at the physical effects of screen time, noting again the correlation between increased screen time and sleep deprivation. Lissak also provided a case study of a nine-year-old male who was diagnosed with ADHD. When the child’s screen time was reduced in the home, several positive effects happened over four weeks, including an increase in physical activity, ability to focus more on schoolwork, less distracted behavior at school and home, and more interest in interacting socially with his family (Lissak, 2018). This case study provided specific evidence presenting the benefits of decreased screen time.

Psychological Effects of Screen Time

Less research has been done on the psychological effects of screen time than on physical effects. However, as screen time has increased in school and at home based on the rise of technology, researchers should continue to look at how children are affected cognitively and emotionally in relation to their screen time.

Domingues (2017) inspects the psychological and physical effects of screen time, stating that beginning with infants and toddlers, screen time can have a long-term effect on early childhood development. The effects of screen time in children two to four years old can have an impact on the development of social skills that are gained in face to face interactions and can increase the risk of low vocabulary and decreased classroom engagement by the time they enter school. Similar studies show that children two to three years old, who had more than three hours of screen time a day, showed increased aggression and peer victimisation by the time they entered school. Domingues (2017) also states that aspects of screen time can have a positive influence, such as the increase in empathy and friendships through social media for children 13-17.

Horowitz-Kraus and Hutton (2017) conducted an experiment that looked at brain connectivity for youth during reading and screen time. The authors expressed concern for the increase of screen time and questioned whether research could show which areas of the brain are active during the time spent in front of devices, such as television or computers. The study shows that children who spend more time reading are able to make connections with visual words to reality. In opposition, screen time was negatively

correlated with cognitive abilities and word connections. Screen time was also negatively correlated with language regions of the brain.

The research for physical and psychological effects based on increased amounts of screen time, usually reported as over two hours a day, show negative effects on children. This is an issue that requires the attention of families, teachers, and legislators in order to preserve the health of young people. The next section of this review looks at research dedicated to screen time in schools. It was difficult to find research related directly to the effects of screen time being used in school.

Screen Time in Schools

Scoggin and Vander Ark (2018) debated whether too much screen time within schools is an issue that needs to be addressed. Each provided a different perspective on the question. “As we sober up from the tech-infused party of the past 20 years, we should think about what should come first in our schools: shaping not just our students’ ability to persevere and solve difficult problems but also their character—their empathic connection with others, their capacity to see our shared humanity, and their ability to problem-solve with others for a common good” (p. 56).

Scoggin (2018) recognizes the benefits of technology in our schools as a tool for teachers and students. However, he also points out the lack of social interaction and social experience that has become apparent in recent years due to increased screen time. His report is concerned about the negative effects that can happen when technology lessens the time students spend in human interaction with one another. Scoggin compares thick and thin institutions stating that “a thick institution becomes part of a

person's identity and engages the whole person: head, hands, heart, and soul" (p. 58).

Increased technology and screen time allow students to disengage from one another and become more anonymous, creating schools that are "thin institutions."

On the opposite side of the issue, Vander Ark (2018) argues that screen time in schools is acceptable as long as it is productive and effective. He explores the positives of using devices in the classroom, including connections to the outside world, increasingly accessible devices, and personalized learning opportunities. Vander Ark admits that the use of screen time depends entirely on the knowledge of the teacher and purposeful use of the device. He explains the benefits of screen time in schools, but also notes that limits are appropriate, especially in young children.

There are many teaching methods that can provide the social interaction that screen time can limit. The next section will explore cooperative learning. This method increases social interaction by giving students the opportunity to work with one another while productively delving into any subject area.

Cooperative Learning

Cooperative learning, also known as collaborative learning, is a style of teaching that gives students an opportunity to work together and learn from one another, usually in small groups. Cooperative learning can be done with any subject and grade level, but has been most popular in the elementary classroom (Slavin, 2015). The research explored in this section highlights the benefits of collaborative learning. This section will look at the most effective elements in a collaborative learning environment, including the benefit of

student interest. It is important to understand the best way to implement cooperative learning in order to enhance cognitive practice, motivate students, and hone social skills.

Successful Implementation of Cooperative Learning

Many studies have been done that show the benefit of cooperative learning in the classroom, indicating improvements in motivation, academic performance, engagement, and behavior. For example, Slavin (2015) concluded that cooperative learning leads to improvement in the following areas:

1. **Motivation.** Students exhibit a high level of motivation to complete tasks when working in a small group. When they value the group's success, students will assist one another and work hard to solve problems.
2. **Social cohesion.** When members of the group cared about the success of their classmates, they were able to help one another and make sure that each member of their group succeeded. Social cohesion can increase through team building activities and self evaluations throughout group time.
3. **Cognitive perspectives.** When working together, students are able to practice learning targets and activities in diverse ways, by teaching others, using vocabulary in conversation, or completing activities with group members. Students are able to cognitively remember material after using it in a variety of ways in their group work.
4. **Structured Group Interaction.** Group interactions are most successful when teachers combine specific strategies for students to work with the learning targets and reward groups for their effort.

According to Slavin (2015) “cooperative learning has the potential to become a primary format used by teachers to achieve both traditional and innovative goals” (p. 15).

An earlier study by Blumenfeld, Marx, Soloway, & Krajcik (1996) also conducted a study to examine the effects of collaborative learning. As with Slavin (2015), Blumenfeld et al (1996) defined elements that create successful small group work, including group norms, accountability, and group composition. Collaboration is described as “a key to help students construct knowledge and to introduce them to disciplinary language, values, and ways of knowing” (p. 39).

Another example of cooperative learning (CL) includes Erdogan (2019) who conducted research to examine seventh-graders using CL and reflective thinking (RT) in a mathematics unit. The study aimed to observe and test students’ critical thinking skills from an experimental and control group. Evidence showed that students involved in CL and RT had a higher rate of success in content and critical thinking skills. Erdogan (2019) notes that “teachers who will apply CL and RT activities in a classroom should have high-level thinking skills such as reflective and critical thinking” (p. 102).

Several more examples of cooperative learning are given in the following section. These studies examine a more specific area of cooperative learning in the subject of CT. The research in the next section is relevant and specific to support the project described in the Methods section of this report.

Learning Computational Thinking “Unplugged”

There have been numerous reports written in recent years in that examine CT lessons taught without the use of computer devices. These lessons are often referred to as

“unplugged.” Most of the following examples have taken place in a secondary education setting. Further research should give additional evidence of the effects of unplugged activities used in the elementary classroom to teach computational thinking.

Paul (2015) highlights the benefits of teaching CT and computer science unplugged. Some examples include sorting and sequencing cards, acting out computer processes, and incorporating physical activities. Paul explains that students who learn CT without computers often feel more motivated, especially those who may not have experience with computers are not held back while learning CT skills. These students can include females and students in poverty, among other demographics.

One study by Peel, Sadler, and Friedrichsen (2019) was conducted in high school classrooms that were learning about natural selection. The lessons used combined standards of natural selection with students exhibiting algorithmic skills. Students presented knowledge by sequencing information, using “if, then” language, and provided algorithms to explain natural selection. These activities did not use devices, but were still able to involve CT at a high level (Peel, Sadler, & Friedrichsen, 2019).

In another study, Sung (2019) described a lesson in which students worked to design a boat that could hold a certain amount of weight (golf balls) before sinking. To aid them, students were given an algorithm to understand the relationship between mass, density, and volume. Computational thinking skills were used to figure out and test the best way to solve the problem. Sung stated that “the unplugged approach has been addressed to teach core concepts of computer science through engaging games, puzzles, or solving real-life problems, using simple computer algorithms” (p. 10).

Another study was conducted by Tarim (2015) in which preschool students practiced pattern recognition, an area of CT, through cooperative learning. In small groups, preschool students worked to identify, extend, and complete patterns. Students worked in small groups in the experimental group. This group outperformed the control group and also exhibited skills of “solidarity, sharing, active listening, and fulfilling their personal responsibilities in the group work activities” (p. 1603).

Limited research has been conducted on the effects of teaching CT using collaborative learning, especially in the elementary environment. The project proposed in the Methods section of this capstone aims to examine teaching algorithmic thinking and pattern recognition, two CT skills, in second grade using collaborative learning methods.

CHAPTER THREE

Project Description

Chapter Overview

Chapter Three provides an overview of the capstone project: writing a unit of lessons to teach second-graders algorithmic thinking and pattern recognition. This chapter describes the unit created, explains the setting and students that the lessons are created for, and presents a brief background of my connection to the setting.

This unit includes lessons that are designed to incorporate collaborative learning, which was described in Chapter Two. The lessons do not use student devices in an effort to minimize screen time used during the school day. The CSTA K-12 Computer Science Standards (2011) are applied to the unit. The goal of the lessons is to help answer the question: *How can second-grade students learn algorithmic thinking and pattern recognition through collaborative learning?*

Curriculum Design

The unit of lessons written for this capstone project follows the *Understanding by Design* model by Wiggins and McTighe (2011). This model creates units and lessons by

using a “backward approach.” The instructor chooses the standards and targets the students should be able to master and works backward to design assessments and activities as part of a three-step process.

The first stage of *Understanding by Design* is to identify the desired results of the unit and lessons (Wiggins & McTighe, 2011). When writing the unit plan, I identified specific learning targets based on standards. This allowed the writing of student activities to be specific and engaging while focusing around a main point or destination. For example, one of the CSTA Computer Science Standard (2017) is “1A-AP-08: model daily processes by creating and following algorithms (sets of step-by-step instructions) to complete tasks” (p. 4).

The second stage of Wiggins and McTighe’s *Understanding by Design* focuses on determining evidence of learning. Once the desired results were identified, I focused on how students would show an understanding of the content. This was achieved through a variety of means, including assessments and observations, that showed mastery of the content.

The final stage of *Understanding by Design* is to design learning activities and instruction (Wiggins & McTighe, 2011). By first determining the learning targets, followed by the assessment, there was a clear idea of what kind of learning activities would lead to an understanding of the lesson content. These activities provided opportunities for students to interact together with the material of the lessons.

Project Framework

For my capstone project, I created a unit of one lesson per week over a six week period. Each lesson is scheduled for 45 minutes and includes an introduction, a student collaborative activity, and a closure around the learning targets. The learning targets for the lessons are based on the following CSTA K-12 Computer Science Standards (2011):

- “Model daily processes by creating and following algorithms (sets of step-by-step instructions) to complete tasks” (p. 4).
- “Debug (identify and fix) errors in an algorithm or program that includes sequences and simple loops” (p. 5).

Students participated in the lessons through collaborative learning activities.

These activities are designed to target the standards and provide opportunities for the students to work together in small groups. Through small group learning, students are able to help one another, learn from each other, and participate in a purposeful, effective conversation about the topic. For example, partners have unique roles that allow students to participate equally while experiencing the skills to lead and to follow directions from others.

The unit plan includes ideas for assessments, learning differences, and directions for successful implementation. For instance, the materials allow for extension to challenge students who have prior knowledge of algorithms and patterns. Each lesson focuses on a new learning target that builds on the previous lesson. The following is a table of lessons and assessments that are provided in the unit plan.

	Learning Target	Assessments	Activities

Lesson One	“I can define the word algorithm.”	Student Pre-Assessment (Interview and Interest Questions)	Students sort what is an algorithm and what is not.
Lesson Two	“I can construct an algorithm to accomplish a task.”	Teacher Observation Form #1	Students build an algorithm for a real-life task.
Lesson Three	“I can construct an algorithm using a code.”		Students build coding algorithms on a grid map.
Lesson Four	“I can revise algorithms to fix errors and missing information.”	Teacher Observation Form #2	Students identify bugs in algorithms and revise them.
Lesson Five	“I can recognize patterns and how they are used in an algorithm.”		Students find patterns and create loops in algorithms.
Lesson	“I can compose	Student Post-Assessment	Students create

Six	an algorithm, using patterns, to achieve a task.”	(Interview and Interest Questions)	algorithms with loops, using a code.
-----	---	------------------------------------	--------------------------------------

The assessments provide a qualitative format to collect written and oral student answers. The student pre and post-assessments represent an interview to question what students know about algorithms and patterns, how they are related to computers, and what the student experience is like to work with these two elements in relation to collaborative learning. Students will be able to define terms, such as algorithm, pattern, loop, and sequence. The activities provide opportunities to demonstrate algorithmic thinking where the students create algorithms, as well as revise algorithms that have errors. Teacher observations provide a chance to note elements of the lesson in which learning through collaboration is working efficiently or lacking. The observation forms provide areas for the teacher to note student mastery, misconceptions, and examples of learning.

Setting and Students

This unit was designed specifically for a second-grade class in an elementary school in a suburb of a large city in the Midwest. The elementary school being used to teach the unit plan, one of ten in the district, provided education to approximately 500 students from grades K-5. The second grade class was split into four sections with each teacher having a class size of about 20.

Students at the school represented a diverse amount of cultures and languages. The school provided 33.1% of students with free or reduced lunch. The student body was made up of 58.1% White, 14.8% Hispanic, 13.3% African American, 8.8% of students of two or more races, 4.3% Asian, and 0.6% American Indian.

Students had some exposure to coding during the school's "Hour of Code" day in two years prior to when this capstone was introduced. Hour of Code, created by code.org, was a push for schools to give K-12 students coding experiences that may pique their interest in computer science or computational thinking. Some of the apps that students have used to code with include: code.org, Kodable, and Scratch, Jr. Though students at the elementary school may have worked with code, they had not received direct instruction on algorithms or pattern recognition in relation to computational thinking.

The classroom of second grade students from the school was an ideal group to expose to the unit plan that was created. The unit plan was an introduction of computational thinking that assisted students through the implementation of computer science in years to come.

Background of Staff

The year the lessons were taught was the first year the school was beginning to implement a "computer science" format to transition the school to a computer science immersion school. The subsequent year was the first year of complete implementation of the immersion program. As part of the transition, teaching staff had received

professional development through both years in computer science and computational thinking.

Staff was also required to be members of the district's Design Thinking Cohort over the two school years of beginning implementation. This cohort focused on expanding elementary students' computational thinking skills. Its goals were to highlight creativity, project-based learning, and elements of computer science. It was the hope of the district and technology staff that this cohort would assist in helping staff create Design Thinking lessons to be used in different content areas.

The school year the capstone project was taught was my fourth year of teaching second-grade in the school described. The unit plan was designed for my own class of students. As part of the technology team and the Design Thinking cohort, I was motivated and excited to teach this unit plan.

Conclusion

Chapter Three described the unit plan that I created to answer the question: *How can second-grade students learn algorithmic thinking and pattern recognition through collaborative learning?* The chapter defined the curriculum design that was used to create the unit and provided a unit framework. It further described the setting and students for which the unit plan will be implemented. Finally, this chapter restated the background of the capstone interest and purpose. In Chapter Four, I provide a conclusion to my project, including how it contributes to the community as public scholarship.

CHAPTER FOUR

Conclusion

Chapter Overview

This capstone project was based on researching the rise of computer science education in schools and questioned whether this increase would result in additional screen time for students. Research around the subject showed the effects of screen time in youth, the push for computational thinking (CT) skills in today's society, and that it is possible for students to learn CT skills without computer devices. The information provided passion to build a unit to teach concepts of algorithmic thinking and pattern recognition to second grade students. The unit was built in order to add on to the CT skills of students without using screen time within the lesson activities. The goal of this unit was to answer the research question: *How can second-grade students learn algorithmic thinking and pattern recognition through collaborative learning?*

Summary of Literature Review

The literature review explored aspects of computational thinking (CT) as a whole and its aspects within education. Several research examples included the categories of algorithms, decomposition, pattern recognition, and abstraction within its key elements that should be taught in relation to CT (Selby & Woollard, 2013). It was evident that algorithmic thinking and pattern recognition would be a successful starting off point for elementary students to study CT.

The review of literature also discussed research around screen time in children, including both psychological and physiological effects shown in relation to increased screen time. Limited research was found that examined screen time within schools, though Scoggin and Vander Ark (2018) compared the positive and negative results of screen time in schools. The literature review provided successful examples of collaborative learning in elementary classrooms. Several studies showed the benefits of collaborative learning, including increased motivation and social cohesion (Slavin, 2015). Relevant research also showed the value of “unplugged” CT lessons. The research combined provided a solid base for building a unit of lessons, using collaborative learning, that focused on two elements of CT: algorithmic thinking and pattern recognition.

Description of the Project

The unit created is comprised of six lessons to build students’ computational thinking skills. The focus of the unit is on algorithmic thinking and pattern recognition.

The lessons provide students with an opportunity to practice computational thinking. Each lesson contains an introduction, activity, and closure that will engage students in practicing skills used by computer scientists. In Lesson One, students work in groups to sort out what an algorithm is and how to define it. Students construct their own basic algorithms in Lesson Two by using examples of real-life tasks. Lesson Three allows students to create an algorithm and test it by following the algorithm on a grid map. Lesson Four allows students to work together to problem-solve algorithms that are incorrect or missing information. Lessons Five and Six build on algorithmic thinking by introducing skills of pattern recognition. These lessons give students experience finding and creating loops within algorithms.

The unit was developed using Wiggins and McTighe (2009) *Understanding by Design*. In the first stage of building the unit, learning targets were selected based on CSTA (2017) standards. Next, student evidence was selected that would show mastery of the learning targets, including a student interest form and teacher observation documents. Finally, collaborative learning activities were created for each lesson to help introduce concepts, engage students, and provide time for students to work with ideas and materials.

Limitations of the Project

This project has some limitations in its current form. Due to its beginning stages, lessons may benefit from further review or experience of other educators. As this lesson was designed for a specific group of students, other teachers may find alterations that

better fit their group of students. Some of the resources used, such as the video in Lesson One, may not be available in all locations over time.

Computer science is an ever-expanding area, and technology is changing by the moment. Thus, the skills of algorithmic thinking and pattern recognition may look different in the future. The materials presented reflect the current trend and skills used by today's computer scientists, but may require changes to stay consistent with the ever-changing computational thinking field.

This unit was written for a specific group of students who had limited experience with computer science content. Other educators aiming to use the unit are encouraged to keep in mind the prior knowledge and backgrounds of each group of students.

Personal Reflection

This capstone has provided an opportunity to build a unit of lessons around a topic for which I am passionate and exuberant. I felt very strongly, as I began choosing my capstone topic and content, that this research should reflect my enthusiasm for computer science and computational thinking (CT) as it has developed in my professional teaching career. The staff at my current school encouraged me to go ahead with my idea, and it felt like kismet that the school was making a push for CT in the curriculum.

Though I enjoy writing my personal opinions and thoughts, it was a new experience writing academically. After five years away from taking graduate courses, it was intimidating stepping into writing a whole capstone. When I began to look for

sources around my topic, it seemed like finding 30 sources was going to be an impossibility. However, as I used scholarly search engines, the world of academia opened up to a vast number of research projects, authors, and topics. It was exciting to become immersed in the subject areas of CT, screen time, and collaborative learning. The idea that my research had a place in the world of CT educational research motivated me to do my best to figure out how to accurately present information and form a relevant and solid project.

It was fitting to work on a unit that applied specifically to my classroom at the same time that computer science was becoming a focus at my own school. The graduate classes at Hamline prepared me with the skills to write detailed lesson plans in all content areas. In the last five years of teaching, I had been exposed to many different curricula and resources which I found helpful when designing this unit. My ideas for layout and verbiage was inspired by materials that I found most helpful and easy to use as an educator.

The work put into this capstone would not have been possible without the attentive support of Hamline professors, my committee, the staff at my school, and my family. Though at times the process was challenging, I am extremely proud of this capstone and hope to use my new-found skills of research and curriculum creation in the future. My goal is that this unit will be used and expanded on with further research, perhaps to include the skills decomposition and abstraction. It would be helpful to work

with other educators who are also passionate about computational thinking in the elementary school.

Conclusion

This unit not only incorporates computational thinking skills, but also amplifies the educational philosophy that “students work better together as a team”. The materials are designed to start students on a lifelong journey to understand and engage with computer science. The research and dedication put into this capstone was for the betterment of young people in an ever-changing, technology-filled world.

REFERENCES

- Angeli, C., Voogt, J., Fluck, A., Webb, M., Cox, M., Malyn-Smith, J., & Zagami, J. (2016). A K-6 computational thinking curriculum framework: implications for teacher knowledge. *Educational Technology & Society*, 19(3), 47+.
- Blumenfeld, P. C., Marx, R. W., Soloway, E., & Krajcik, J. (1996). Learning With Peers: From Small Group Cooperation to Collaborative Communities. *Educational Researcher*, 25(8), 37–39. <https://doi.org/10.3102/0013189X025008037>
- Bureau of Labor Statistics, U.S. Department of Labor (2016). Computer and Information Research Scientists : Occupational Outlook Handbook. Retrieved from <https://www.bls.gov/ooh/computer-and-information-technology/computer-and-information-research-scientists.htm>
- CSTA and ISTE (2011). Computational Thinking in K–12 Education leadership toolkit. Retrieved from <http://csta.acm.org/Curriculum/sub/CurrFiles/471.11CTLeadershipToolkit-SP-vF.pdf>.

- CSTA K-12 Computer Science Standards. (2017). *Computer Science Teachers Association (CSTA)*. Retrieved from [https://www.doe.k12.de.us/cms/lib/DE01922744/Centricity/Domain/176/CSTA Computer Science Standards Revised 2017.pdf](https://www.doe.k12.de.us/cms/lib/DE01922744/Centricity/Domain/176/CSTA%20Computer%20Science%20Standards%20Revised%202017.pdf).
- Denning, P. J. (2009). Beyond Computational Thinking. *Communications of the ACM*, 52(6), 28-30. doi:10.1145/1516046.1516054
- Domingues, M. S. (2017). Clinical and psychological effects of excessive screen time on children. *Journal of Paediatrics and Child Health*, 53(4), 333–338. <https://doi-org.ezproxy.hamline.edu/10.1111/jpc.13462>
- Ellison, C. M., Boykin, A. W., Tyler, K. M., & Dillihunt, M. L. (2005). Examining Classroom Learning Preferences among Elementary School Students. *Social Behavior & Personality: An International Journal*, 33(7), 699–708. <https://doi-org.ezproxy.hamline.edu/10.2224/sbp.2005.33.7.699>
- Erdogan, F. (2019). Effect of Cooperative Learning Supported by Reflective Thinking Activities on Students' Critical Thinking Skills. *Eurasian Journal of Educational Research*, (80), 89–112. Retrieved from <https://search-ebshost-com.ezproxy.hamline.edu/login.aspx?direct=true&db=eric&AN=EJ1211625&site=ehost-live>

- Foltz, J. L., Cook, S. R., Szilagyi, P. G., Auinger, P., Stewart, P. A., Bucher, S., Baldwin, C. D. (2011). US Adolescent Nutrition, Exercise, and Screen Time Baseline Levels Prior to National Recommendations. *Clinical Pediatrics*, 50(5), 424–433. <https://doi.org/10.1177/0009922810393499>
- Futschek, G. (2006). Algorithmic thinking: The key for understanding computer science. In Informatics education—the bridge between using and understanding computers (pp. 159–168). Berlin: Springer.
- Gingold, J. A., Simon, A. E., & Schoendorf, K. C. (2014). Excess Screen Time in US Children: Association With Family Rules and Alternative Activities. *Clinical Pediatrics*, 53(1), 41–50. <https://doi.org/10.1177/0009922813498152>
- Gürbüz, H., Evlioglu, B., Erol, Ç. S., Gülseçen, H., & Gülseçen, S. (2017). “What’s the Weather Like Today?”: A Computer Game to Develop Algorithmic Thinking and Problem Solving Skills of Primary School Pupils. *Education and Information Technologies*, 22(3), 1133–1147.
- Horowitz-Kraus, T., & Hutton, J. S. (2017). Brain connectivity in children is increased by the time they spend reading books and decreased by the length of exposure to screen-based media. *Acta Paediatrica*, 107(4), 685-693. doi:10.1111/apa.14176

Kalelioğlu, F., Kukul, V., & Gülbahar, Y. (2016). A Framework for Computational Thinking Based on a Systematic Research Review. *Baltic J. Modern Computing*, 4(3), 583-596. Retrieved from https://www.bjmc.lu.lv/fileadmin/user_upload/lu_portal/projekti/bjmc/Contents/4_3_15_Kalelioglu.pdf.

Kazimoglu, C., Kiernan, M., Bacon, L., & Mackinnon, L. (2012). A Serious Game for Developing Computational Thinking and Learning Introductory Computer Programming. *Procedia - Social and Behavioral Sciences*, 47, 1991-1999. doi:10.1016/j.sbspro.2012.06.938

Krieger, L. M. (2019, April 2). U.S. College Students Excel in Computer Science. *Emergency Management*. Retrieved from <https://search-ebshost-com.ezproxy.hamline.edu/login.aspx?direct=true&db=rps&AN=2W62008784777&site=ehost-live>

Lee, I., Martin, F., Denner, J., Coulter, B., Allan, W., Erickson, J., Werner, L. (2011). Computational thinking for youth in practice. *ACM Inroads*, 2(1), 32-37. doi:10.1145/1929887.1929902

Lissak, G. (2018). Adverse physiological and psychological effects of screen time on children and adolescents: Literature review and case study. *Environmental Research, 164*, 149–157.

<https://doi-org.ezproxy.hamline.edu/10.1016/j.envres.2018.01.015>

Minnesota Department of Education (2011). *2011 Minnesota K-12 Academic Standards in Social Studies*. Retrieved from <https://education.mn.gov/MDE/dse/stds/soc/>

Ozturk, Z., Dooley, C. M., & Welch, M. (2018). Finding the Hook: Computer Science Education in Elementary Contexts. *Journal of Research on Technology in Education, 50*(2), 149-163. doi:10.1080/15391523.2018.1431573

Paul, A. M. (2015). Teaching Computer Science— Without Touching a Computer. *Education Digest, 80*(5), 23-26. Retrieved from <http://www.eddigest.com>

Peel, A., Sadler, T. D., & Friedrichsen, P. (2019). Learning natural selection through computational thinking: Unplugged design of algorithmic explanations. *Journal of Research in Science Teaching*.
<https://doi-org.ezproxy.hamline.edu/10.1002/tea.21545>

Price, C. B., & Price-Mohr, R. M. (2018). An Evaluation of Primary School Children Coding Using a Text-Based Language (Java). *Computers in the Schools, 35*(4),

284-301. doi:10.1080/07380569.2018.1531613

Robinson, T. N., & Borzekowski, D. L. (2006). Effects of the SMART Classroom Curriculum to Reduce Child and Family Screen Time. *Journal of Communication*, 56(1), 1-26. doi:10.1111/j.1460-2466.2006.00001.x

Scoggin, D., & Vander Ark, T. (2018). Should We Limit "Screen Time" in School? *Education Next*, 18(1), 54-63. Retrieved from educationnext.org

Selby, C., & Woollard, J. (2013). *Computational thinking: The developing definition*. Retrieved from <https://eprints.soton.ac.uk/356481/>

Slavin, R. E. (2015). "Cooperative Learning in Elementary Schools." *Education 3–13* 43(1): 5–14. doi:10.1080/03004279.2015.963370

Sung, E. (2019). Fostering computational thinking in technology and engineering education: An unplugged hands-on engineering design approach. *Technology & Engineering Teacher*, 78(5), 8-13. Retrieved from <http://www.iteea.org/>

Tarim, K. (2015). Effects of Cooperative Group Work Activities on Pre-school Children's Pattern Recognition Skills. *Educational Sciences: Theory & Practice*, 15(6). doi:10.12738/estp.2016.1.0086

Tran, Y. (2019). Computational Thinking Equity in Elementary Classrooms: What Third-Grade Students Know and Can Do. *Journal of Educational Computing Research*, 57(1), 3–31. <https://doi.org/10.1177/0735633117743918>

The American Heritage® Science Dictionary. (2011). Houghton Mifflin Harcourt Publishing Company. Retrieved from <https://www.dictionary.com/browse/computer-science>

Wing, J. M. (2006). Computational Thinking. *COMMUNICATIONS OF THE ACM*, 49(3), 33-35. Retrieved from <https://dl.acm.org/citation.cfm?id=1118215>.

Yadav, A., Hong, H., & Stephenson, C. (2016). Computational Thinking for All: Pedagogical Approaches to Embedding 21st Century Problem Solving in K-12 Classrooms. *TechTrends*, 60(6), 565-568. doi:10.1007/s11528-016-0087-7

Yadav, A., Krist, C., Good, J., & Caeli, E. N. (2018). Computational thinking in elementary classrooms: Measuring teacher understanding of computational ideas for teaching science. *Computer Science Education*, 28(4), 371-400. doi:10.1080/08993408.2018.1560550

APPENDIX A

Lesson Plans

Lesson Plans and Materials

1. Introduction to Algorithms

Lesson Plan

Student Interest Form #1

“Algorithm or Not” Sorting Cards

2. Constructing Algorithms

Lesson Plan

Teacher Algorithm Creation Example

Algorithm Creation Sheet

Teacher Observation Form #1

3. Following Algorithms

Lesson Plan

Practice Algorithm Packet

Practice Algorithm Board

4. Revising Algorithms

Lesson Plan

Teacher Revising Example

Revising Algorithms Packet

Teacher Observation Form #2

5. Patterns in Algorithms

Lesson Plan

Teacher Loop Example

Loop Packet

6. Composing a Detailed Algorithm

Lesson Plan

My Finest Algorithm - Choice A

My Finest Algorithm - Choice B

My Finest Algorithm - Answer Key

Student Interest Form #2

Topic: Computational Thinking**Lesson Title:** Introduction to Algorithms

Lesson Title	Introduction to Algorithms (Lesson 1 out of 6)
Content Standards	CSTA Standard 1A-AP-08 Model daily processes by creating and following algorithms (sets of step-by-step instructions) to complete tasks.
Content Objective	I can define the word algorithm.
Academic Language	Computational Thinking, Computer Scientist, Algorithm
Sentence Starters	This is/is not an algorithm. I know because... I agree because... I disagree because... An algorithm is...
Assessment	Formal: Students will fill out a pre-assessment and interest form. Informal: Teacher will observe students learning during the activity portion of the lesson.
Provisions for Learning Differences	Pre-assessment may be given verbally for students who are not able to read it on their own. Form groups of differing levels of ability.
Materials	Student Interest Form # 1 (Pre-Assessment) Algorithm Sorting Cards Teacher Computer and Display Device
Learning Activities	
Engage/Instruction (15 mins)	Explain that students will begin a six week exploration of how computers “think” or “process” information. This is called “computational thinking.” Students will get a chance to practice being computer scientists by thinking like a computer and learning computational thinking skills. Whole class will brainstorm examples of tasks computers that can do. Examples include: create documents, print reports, solve math equations, make phone calls, play music, etc. Record class ideas on chart paper.

	<p>Explain that the target of today’s lesson is to define the word “algorithm.” Share that computers follow an algorithm to complete tasks.</p> <p>Show video of algorithm explanation. https://www.youtube.com/watch?v=Da5TOXCwLSg</p>
<p>Activity (20 mins)</p>	<p>After the video, introduce activity: “What is an algorithm?”</p> <p>Directions:</p> <ol style="list-style-type: none"> 1. Students will be put into groups of 3-4. 2. Groups are given “Algorithm or Not” cards. 3. Students will discuss with one another and sort whether the example is an algorithm or not an algorithm. 4. Students may use sentence starters provided. All group members must agree on which activity is an algorithm. <p>After groups have sorted most of the cards. Groups will share out which items have been sorted into which categories.</p> <p>Each group will come up with their definition of what is an algorithm. Facilitate discussion and take notes on chart paper as groups share out their definition. The definition should come out similar to “a step-by-step set of instructions to complete a task.”</p> <p>Students will look back at their cards and see if they need to resort cards to fit the new definition.</p> <p>Finally, students will collect materials and turn in cards.</p> <p>**Important understandings to note: Algorithms can be followed by humans, not just computers. They can be given verbally or written with words, pictures, or numbers.</p>
<p>Closure (10 mins)</p>	<p>Review the class’s definition of algorithm. Explain that in the next lesson will be able to build their own algorithms.</p> <p>Next, show the pre-assessment/interest form. Explain how to fill it out and ask students to be honest about their feelings, noting that students do not have to know all the answers.</p> <p>Students will have time to fill out pre-assessment on their own and turn-in to teacher.</p>

Student Interest Form #1

1. Working on computers makes me feel...

2. What I know about how computers work...










4. Match the term with its definition.

(It's ok if you don't know. Just do your best.)

Algorithm	A certain order of events or things
Pattern	A way of completing tasks and solving problems
Computational Thinking	A step-by-step series of instructions to complete a task
Loop	The study of how computers work
Sequence	A series that is repeated
Computer Science	An instruction that represents a series of steps

5. What would you like to learn about computational thinking?

“Algorithm or Not?” Sorting Cards

<p>Directions to a Friend's House</p> 	<p>A Recipe for Pancakes</p> 	<p>Watching Television</p> 
<p>Steps for How to Kick a Soccer Ball</p> 	<p>Pulling out a Random Card from a Deck</p> 	<p>Explaining the Movements to do a Cartwheel</p> 
<p>The Wind Blowing Dandelion Seeds</p> 	<p>How a Calculator Solves a Math Problem</p> 	<p>Guide for How to Change a Lightbulb</p> 

Topic: Computational Thinking**Lesson Title:** Constructing Algorithms

Lesson Title	Introduction to Algorithms (Lesson 2 out of 6)
Content Standards	CSTA Standard 1A-AP-08 Model daily processes by creating and following algorithms (sets of step-by-step instructions) to complete tasks.
Content Objective	I can construct an algorithm to accomplish a task.
Academic Language	Computational Thinking, Computer Scientist, Algorithm
Sentence Starters	An algorithm is ... My first step is ... AL would need to know ...
Assessment	Formal: Teacher Observation Form #1 Informal: Algorithm Creation Sheet
Provisions for Learning Differences	Students may write or draw each step of the algorithm. Partners should be of differing abilities.
Materials	Observation Form # 1 Algorithm Creation Sheet Algorithm Creation Sheet Example
Learning Activities	
Engage/Instruction (10 mins)	Begin the lesson by introducing AL the Alien. AL can speak English and understand most words, but he's new to earth and doesn't know basic skills like how to provide for himself with earthly things. If you told AL to make himself a bowl of cereal: Would he be able to do it? Would it go smoothly or would there be a mess? Share with a partner what might happen. (He might not know how much to put in the bowl. He may spill the milk all over. He might not know how to eat it with a spoon.) After students share, ask them "what might help AL to do this without knowing what it's supposed to look like?" Students should come up with ideas, such as "give him step-by-step instructions or an algorithm." If they do not, suggest an algorithm to help.

	<p>Review the class's definition of algorithm. Ask students to recall some examples of algorithms from Lesson 1.</p> <p>Next, explain that students are going to be computer scientists today and build their own algorithm for AL, just like professionals build for computers.</p>
<p>Activity (30 mins)</p>	<p>Give directions for students with the Algorithm Creation Sheet. They will show Algorithm Creation Example Sheet to show how to fill a water bottle.</p> <ol style="list-style-type: none"> 1. Unscrew cap. 2. Set cap aside. 3. Turn on faucet. 4. Pick up water bottle. 5. Place water bottle opening under the faucet. 6. When bottle is full, remove from faucet. 7. Place water bottle down on counter. 8. Turn off faucet. 9. Screw cap back on water bottle. <p>Note that AL needs really specific instructions. If you just said "fill water bottle," he will not know how to do that.</p> <p>Pair students to create their algorithm with a partner. Students may use words or pictures to make their algorithm.</p> <p>Choices for Algorithms: How to put on jacket. How to make a peanut butter and jelly sandwich. How to draw a smiley face. Challenging Option: How to tie your shoes. (Other teacher approved options can be used.)</p> <p>**During this time, walk around the room to check in with students and fill out Observation Form #1.</p> <p>After students have completed their algorithm, they should join another pair to share their algorithm. Remind students that computer scientists work together, so students should feel free to suggest improvements and use each other's ideas.</p>

	<p>Pull the whole group together to share a couple examples with the whole class. (Try to select students who may not be recognized as much in other content areas.)</p>
<p>Closure (5 mins)</p>	<p>Congratulate students on creating algorithms!</p> <p>Explain that while students used words and pictures, a computer scientist will use code or “computer language” to give a computer steps to complete a task.</p> <p>Let them know that next week students will get a chance to use a code to help AL accomplish more tasks.</p>

Name _____

Teacher **EXAMPLE** Algorithm Creation

Create an algorithm (step by step directions) for AL the Alien to complete a task. Choose an example from the list and write or draw steps. Use the back of this paper if necessary. Be specific! AL needs to know each step!

Choices for you Algorithm:

How to put on jacket.

How to make a peanut butter and jelly sandwich.

How to draw a smiley face.

Challenging Option: How to tie your shoes.



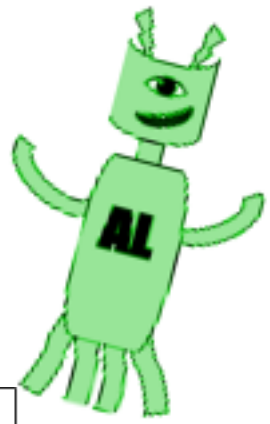
Task I Chose for AL: How to fill a water bottle.

Directions for AL the Alien		(Write steps or draw a picture of each step.) Example: Get out two pieces of bread.	
1.	Unscrew cap of bottle.	2.	Set cap aside on counter.
3.	Turn on the faucet of the sink.	4.	Pick up water bottle.
5.	Place the water bottle opening under the faucet.	6.	When the bottle is full, remove it from the faucet.
7.	Place water bottle on the counter.	8.	Turn off the faucet.
9.	Screw the cap back on the water bottle	10	.

Name _____

Algorithm Creation Sheet

Create an algorithm (step by step directions) for AL the Alien to complete a task. Choose an example from the list and write or draw steps. Use the back of this paper if necessary. Be specific! AL needs to know each step!



Choices for you Algorithm:

- How to put on jacket.
- How to make a peanut butter and jelly sandwich.
- How to draw a smiley face.
- Challenging Option: How to tie your shoes.

Task I Chose for AL:

Directions for AL the Alien		(Write steps or draw a picture of each step.) Example: Get out two pieces of bread.	
1.		2.	
3.		4.	
5.		6.	
7.		8.	
9.		10	
		.	

Topic: Computational Thinking**Lesson Title:** Following Algorithms

Lesson Title	Following Algorithms (Lesson 3 out of 6)
Content Standards	CSTA Standard 1A-AP-08 Model daily processes by creating and following algorithms (sets of step-by-step instructions) to complete tasks.
Content Objective	I can construct and follow an algorithm using a code.
Academic Language	Computational Thinking, Computer Scientist, Algorithm, Coding
Sentence Starters	Coding is ... OR A code is ... The directions for the algorithm is ...
Assessment	Formal: None. Informal: Teacher will observe students learning during the activity portion of the lesson. Students will complete a Practice Algorithm Packet.
Provisions for Learning Differences	Let students help each other in groups. Groups should have a variety of learners.
Materials	Teacher Example Practice Algorithm Packet Practice Algorithm Board (Laminated with Pieces Cut Out) Dry Erase Boards Dry Erase Markers
Learning Activities	
Engage/Instruction (10 mins)	Have students gather and turn to a partner to redefine algorithm and talk about what they did in the previous lesson. (Writing an algorithm.) Explain that computers are similar to AL the Alien, in that they need specific directions. However, computer programmers can only use a limited number of ways to direct the computer. Explain the term code (instructions for the program). Today, we will practice using algorithms to guide AL around a map.

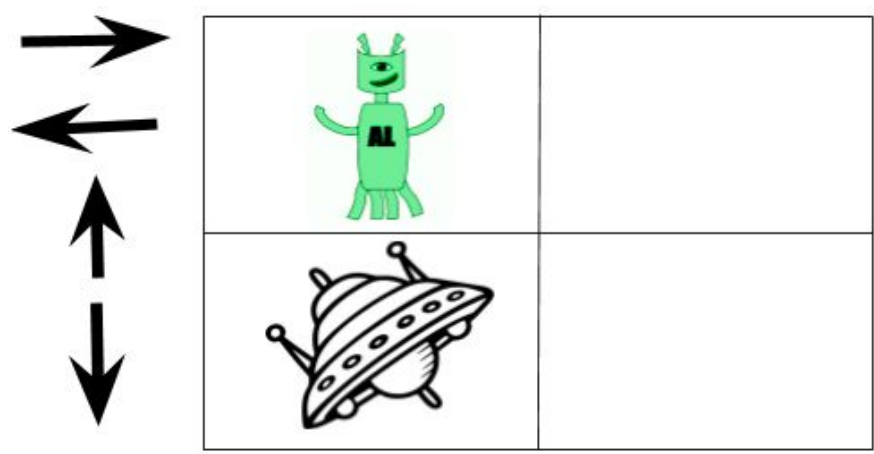
	<p>This time AL the Alien has to get to his spaceship, but you may only use arrows for him to move. This is like how a computer only reads lines of code.</p> <p>Show Teacher Example of what AL should do to get to the spaceship. Get class input as to what directions to give AL for first three activities. Demonstrate that they will have to practice the pattern to make sure that it is correct for AL.</p>
<p>Activity (30 mins)</p>	<p>Next, students will be able to work on their own to complete lines of code in their Practice Algorithm Packet. Students should sit near a group of 4-5 and compare the pages. If they need help, they can ask someone in their group.</p> <p>Once they have completed the first few tasks, students can create their own example of code with the Practice Algorithm Board. Here they can place the spaceship, put AL on a space on the map, and challenge a partner to solve their code. They should write their code on a separate sheet of paper or dry erase board.</p> <p>Encourage students who are grasping the concept to make the activity more challenging by creating more barriers or limiting their partner to a certain number of moves.</p>
<p>Closure (5 mins)</p>	<p>Have students clean up and gather together. Ask students to share with the class what they enjoyed about the activity. Then ask what was challenging or surprising.</p> <p>Let students know that the next lesson they will be working with algorithms that have mistakes. Part of a computer scientist's job is to figure out errors and revise algorithms.</p>

Name _____

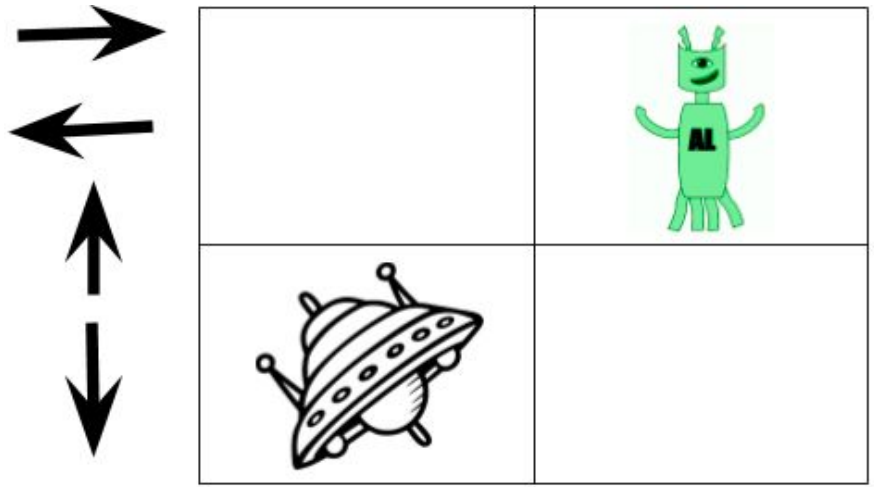
Practice Algorithm Packet

Follow the directions for each activity to practice coding. You are on your way to becoming an Algorithm Expert!

1. Circle the arrow that will get AL the Alien to his spaceship.





2. Circle the **TWO** arrows that will get AL the Alien to his spaceship.





**Check with a partner or your group to see if you can help one another or share ideas.

3. Now, **DRAW** the arrows that will get AL the Alien to his spaceship. You can use arrows more than once.

→
←
↑
↓

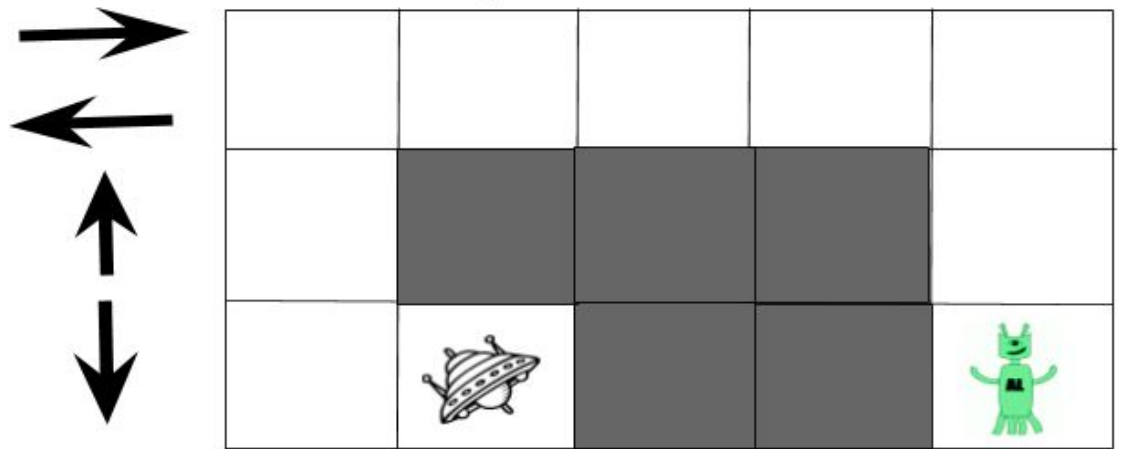
4. **DRAW** arrows that will get AL the Alien to his spaceship. You can use arrows more than once.

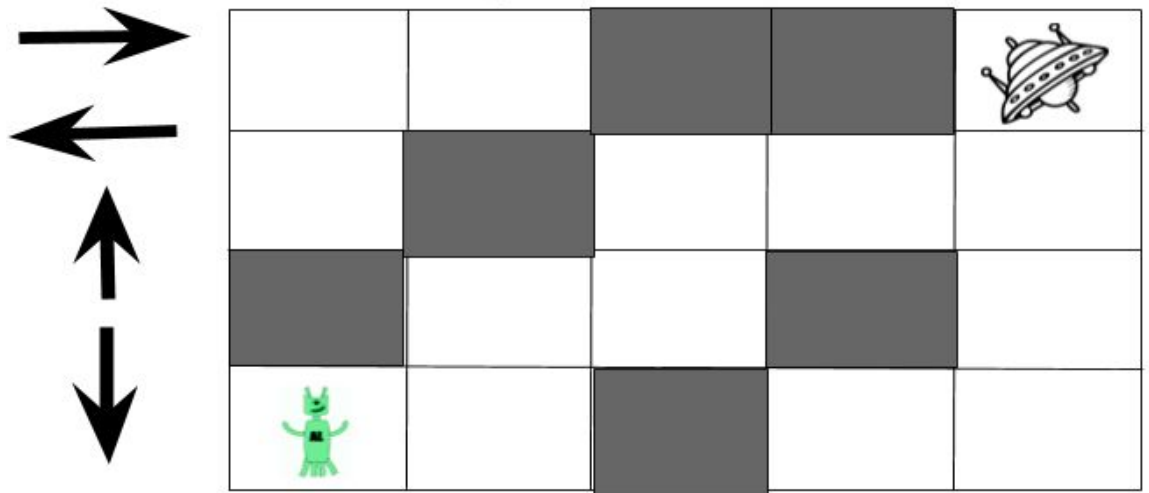
→
←
↑
↓

**Check with a partner or your group to see if you can help one another or share ideas.

5. **DRAW** arrows that will get AL the Alien to his spaceship.
AL **CANNOT** go over dark blocks.

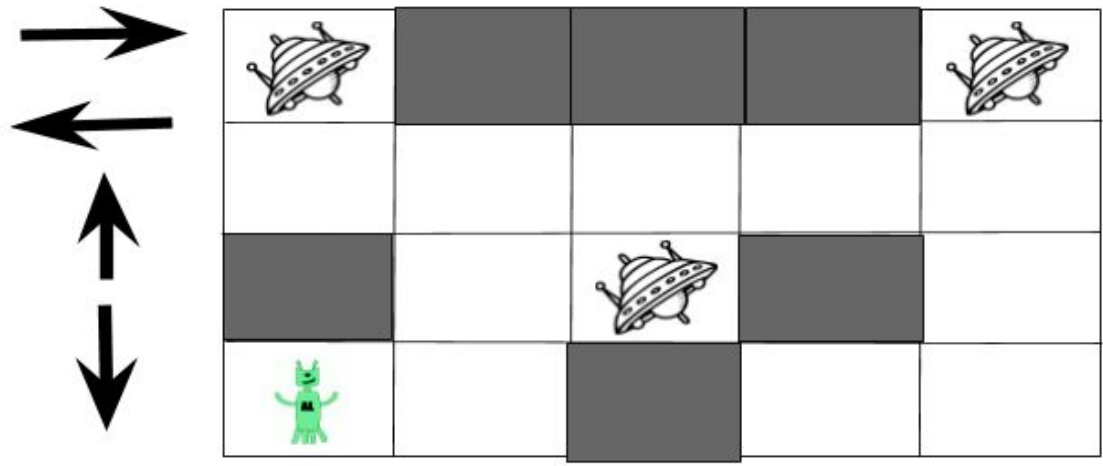


6. **DRAW** arrows that will get AL the Alien to his spaceship.
AL **CANNOT** go over dark blocks.



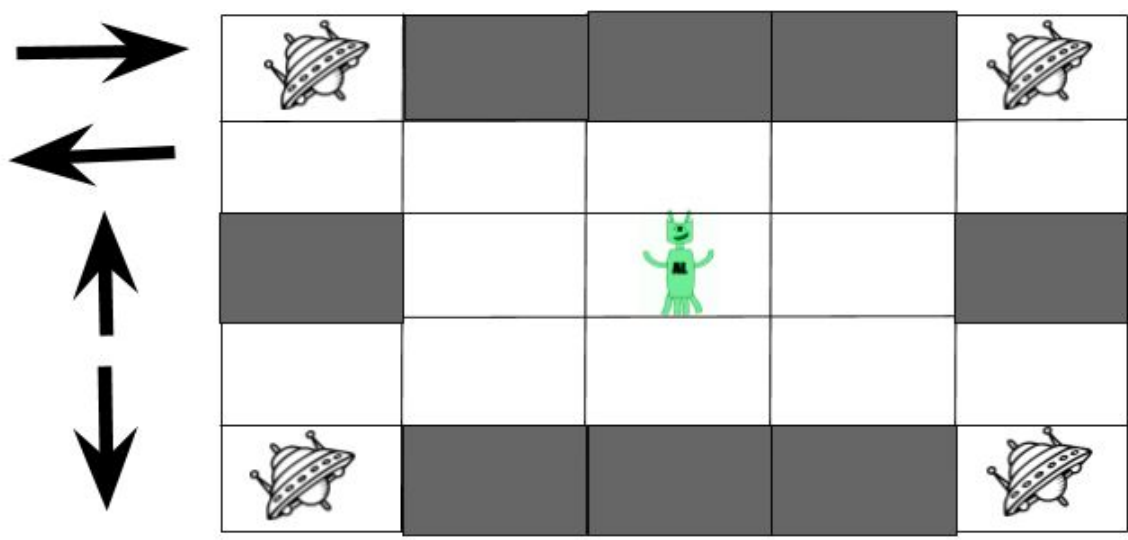
7. **DRAW** arrows that will get AL to all go over each of his spaceships.

AL **CANNOT** go over dark blocks.



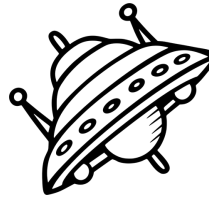
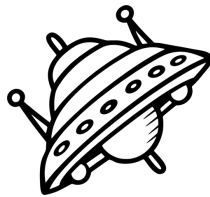
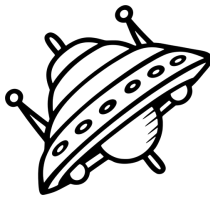
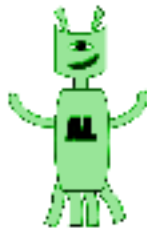
8. **DRAW** arrows that will get AL to all go over each of his spaceships.

AL **CANNOT** go over dark blocks.



Practice Algorithm Board

*Fill in blocks for barriers with a dry erase marker.



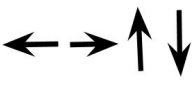
Topic: Computational Thinking**Lesson Title:** Revising Algorithms

Lesson Title	Revising Algorithms (Lesson 4 out of 6)
Content Standards	<p>CSTA Standards</p> <p>1A-AP-08</p> <p>Model daily processes by creating and following algorithms (sets of step-by-step instructions) to complete tasks.</p> <p>1A-AP-14</p> <p>Debug (identify and fix) errors in an algorithm or program that includes sequences and simple loops.</p>
Content Objective	I can revise algorithms to fix errors and missing information.
Academic Language	Computational Thinking, Computer Scientist, Algorithm Bug, Debug, Revise
Sentence Starters	This algorithm has a bug. I can debug it by ... I can revise this algorithm by ...
Assessment	<p>Formal: Teacher will fill out Teacher Observation Form #2.</p> <p>Informal: Students will fill in the Revising Algorithms Packet.</p>
Provisions for Learning Differences	Students may write or draw each step of the algorithm. Partners should be of differing abilities.
Materials	Teacher Revising Example Revising Algorithms Packet Teacher Observation Form #2
Learning Activities	
Engage/Instruction (10 mins)	<p>Introduce a Teacher Revising Example to students. Move AL along the sequence that is set up. When AL does not get to where he needs to be, ask them what happened. (There's a mistake in the code.)</p> <p>Explain the terms "bug" (a mistake in a code) and "debug" (fixing a code that contains a mistake). Tell students that they will have to</p>

	<p>follow codes today to find out where the “bug” is and figure out how to “debug” the code.</p> <p>Use Teacher Revising Example to point out the bug in the code and revise or fix it to complete the task.</p>
<p>Activity (30 mins)</p>	<p>Students will work in their group today to find the bugs in the code and debug them to complete the tasks. They will be using a new example of coding.</p> <p>Demonstrate how the coding works for this system. Next, distribute Practice Algorithm Packet #2. The packets can be completed as partners. Ask students to switch between being a WRITER and a TALKER. The WRITER should touch the packet and write down ideas, and then they will switch for each task. This will encourage students to try out both roles of leading and writing down the debugging.</p> <p>If students finish the packet, they can check with their group members. Then they can create a code on the back with a bug for other group members to correct.</p>
<p>Closure (5 mins)</p>	<p>Ask students to share bugs and debugging examples from their packets. Students can check their packets to see if they solved the codes in other ways. Note that computer scientists need to be creative and sometimes there is more than one solution to a problem!</p>

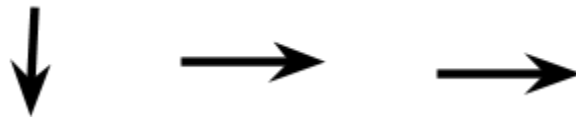
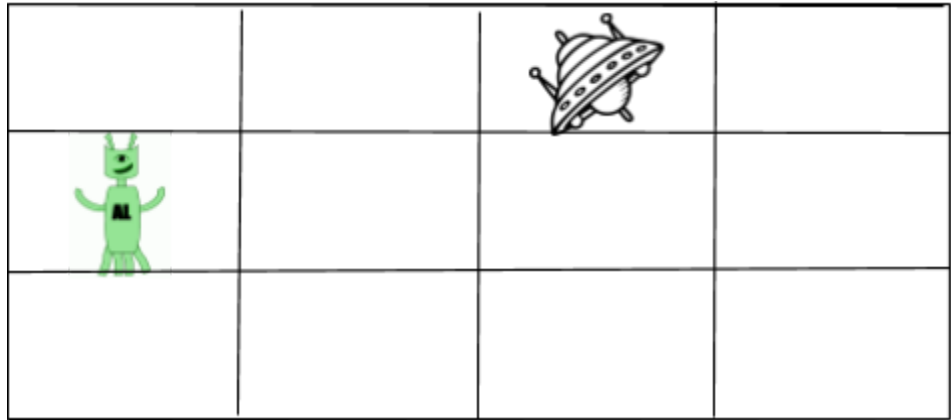
Teacher Revising Example

Add or cross out directions to DEBUG the algorithms.

Remember to only use these arrows: 

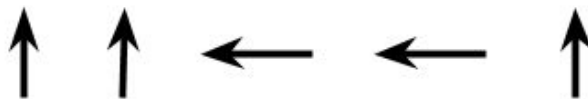
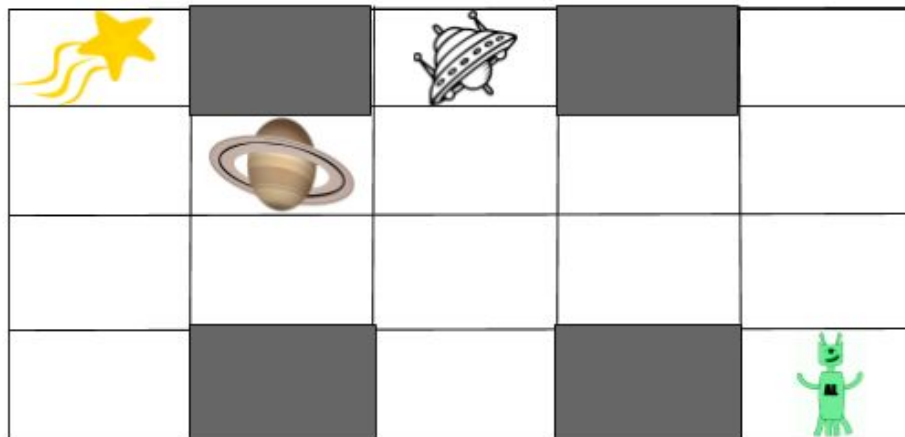
Example 1: FOLLOW the algorithm.

Is there a BUG? If so, how can you REVISE or FIX it?



Example 2: FOLLOW the algorithm.

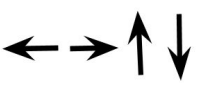
AL wanted to get to  . Where is the bug?



Name _____



Revising Algorithms Packet

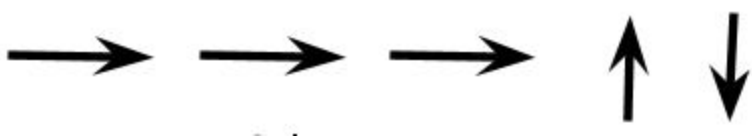
Add or cross out directions to DEBUG the algorithms.

Remember to only use these arrows: 

1. FOLLOW the algorithm.



Is there a BUG? If so, how can you REVISE or FIX it?



2. FOLLOW the algorithm.

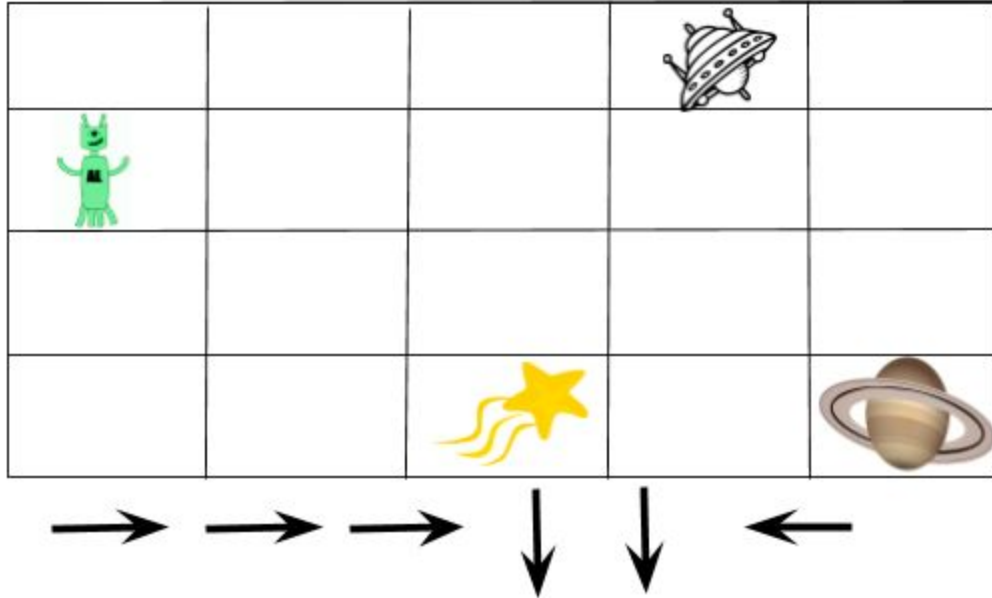
Is there a BUG? If so, how can you REVISE or FIX it?



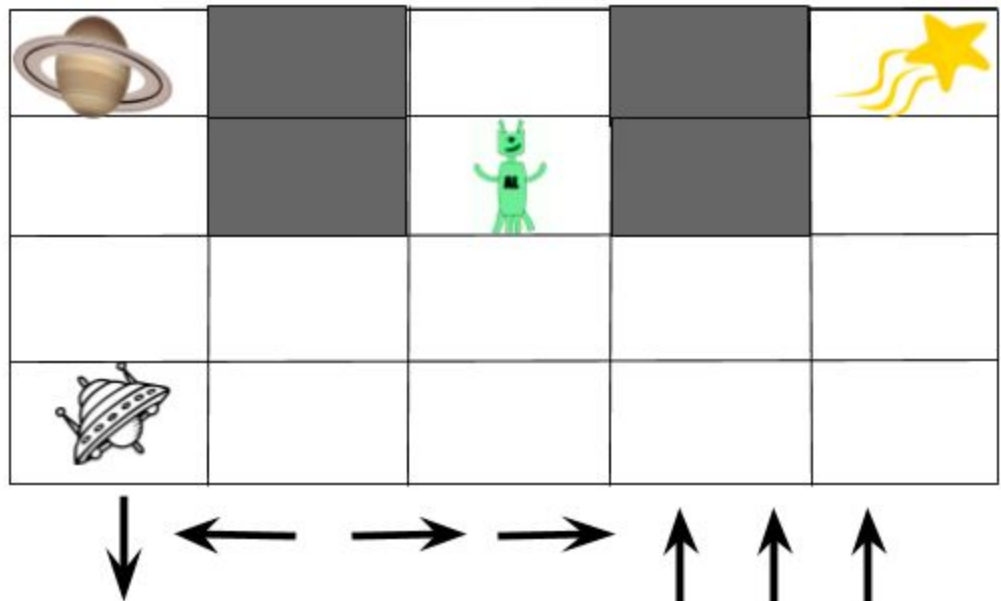
3. FOLLOW the algorithm.

AL wanted to get to  . Where is the bug?



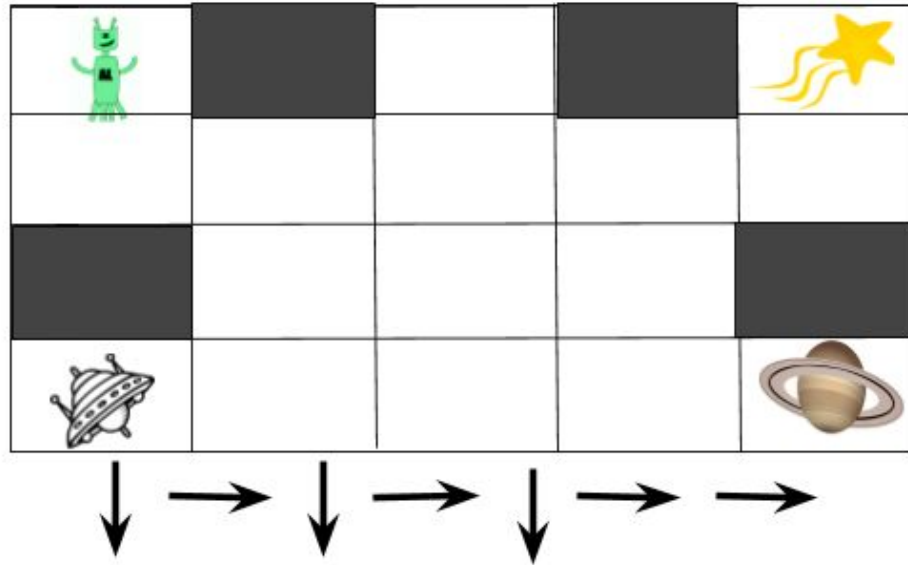
4. FOLLOW the algorithm.

AL wanted to get to  . Where is the bug?





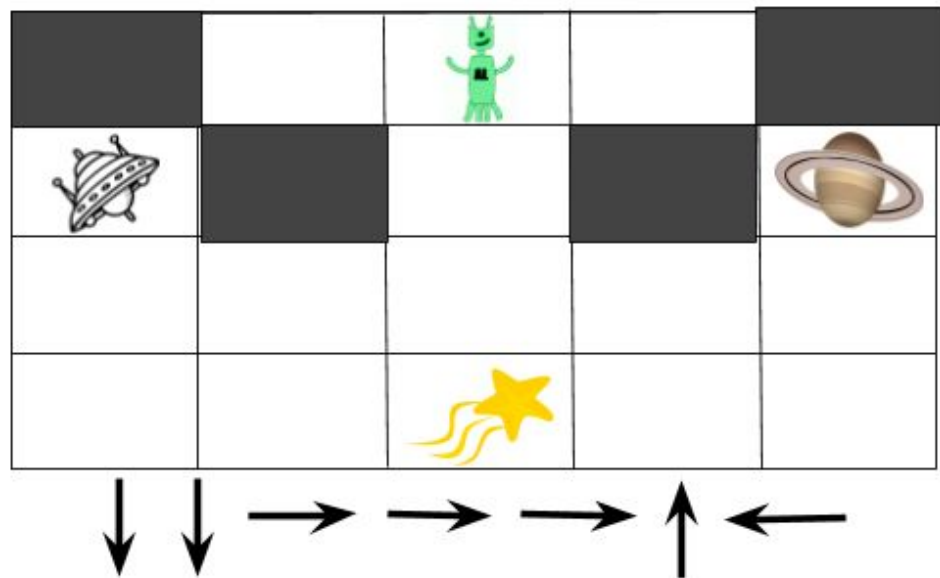
5. FOLLOW the algorithm.

AL wanted to get to  . Where is the bug?
(More than one arrow may need debugging.)



6. FOLLOW the algorithm.

AL wanted to get to , then to  .
Where is the bug?
(More than one arrow may need debugging.)



Topic: Computational Thinking**Lesson Title:** Patterns in Algorithms

Lesson Title	Patterns in Algorithms (Lesson 5 out of 6)
Content Standards	<p>CSTA Standards</p> <p>1A-AP-08</p> <p>Model daily processes by creating and following algorithms (sets of step-by-step instructions) to complete tasks.</p> <p>1A-AP-14</p> <p>Debug (identify and fix) errors in an algorithm or program that includes sequences and simple loops.</p>
Content Objective	I can revise algorithms to fix errors and missing information.
Academic Language	Computational Thinking, Computer Scientist, Algorithm, Loop
Sentence Starters	A loop is ... This is a loop because ...
Assessment	<p>Formal: Teacher will fill out Teacher Observation Form #2.</p> <p>Informal: Students will fill in the Revising Algorithms Packet.</p>
Provisions for Learning Differences	
Materials	<p>Teacher Observation Form # 2</p> <p>Loop Packet</p>
Learning Activities	
Engage/Instruction (10 mins)	<p>Remind students that have worked on defining algorithm, writing algorithms, and debugging already written algorithms.</p> <p>Introduce the new term: LOOP. Explain that a LOOP is a pattern of movements. Let students talk to each other about what this means. Together, use the Teacher Loop Example to locate patterns in the algorithms provided. (Example Answer: <u>Loop 1, Loop 1, X</u>)</p> <p>Next, explain that they will be using a new form of algorithm building. Describe the character, Inky, who has a new function. She</p>

	<p>can color in a square with the FILL command. Their goal is to create algorithms for Inky to use, using loops (patterns). Give them an example of how to solve problem number one in the pattern packet. Students will work with a partner, using WRITER and TALKER roles.</p>
<p>Activity (30 mins)</p>	<p>Hand out the Loop Packets. Partners should first work on identifying patterns and how to write them as a LOOP.</p> <p>Students will work as WRITERS and TALKERS on their packets. They will need to listen to each other and follow the rules of WRITER and TALKER.</p> <p>They should figure out how to get Inky to travel and FILL blocks to replicate the image given. If they need help, they should ask other partners. Each time they switch to a new activity, they should switch roles.</p> <p>Finally, students may use Loop Boards OR Practice Algorithm Boards with dry erase markers to come up with algorithms that include loops.</p> <p>Observe students and take notes, if necessary, for how students are able to reach the target, misconceptions, and how students work together.</p>
<p>Closure (5 mins)</p>	<p>Ask students for some examples of LOOPS in their packet. Share with the class. Ask students what they noticed while they tried to use LOOPS with their algorithms.</p> <p>Tell students that in the final lesson they will be able to write algorithms with loops and be able to debug complex algorithms.</p>

Teacher Loop Example

Identify the PATTERN in each algorithm. Next, create a loop and rewrite the algorithm.


Inky has an extra function for her algorithms:



← → ↑ ↓ AND X

X fills in squares with ink.

(You may color in squares with a pencil or crayon to represent this.)

X → ↓ X → ↓ X

Circle the repeating actions.

Label them as Loop 1.

Loop 1 =

Now write out your algorithm with Loop 1 as a direction.

Name _____

Loop Packet

Identify the PATTERN in each algorithm. Next, create a loop and rewrite the algorithm.


Inky has an extra function for her algorithms:

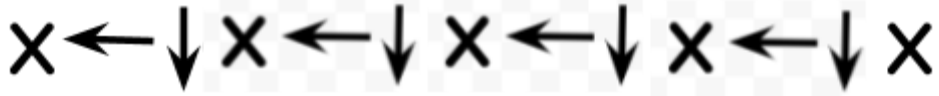


X fills in squares with ink.

(You may color in squares with a pencil or crayon to represent this.)

1. Circle the pattern. Then, complete the algorithm. What did you make?




2. Circle the pattern.

Next, define Loop.

Finally, complete the algorithm.

What did you make?

X → X → X → X → X →
 ← ← ↓ X ↓ X ↓ X ↓ X

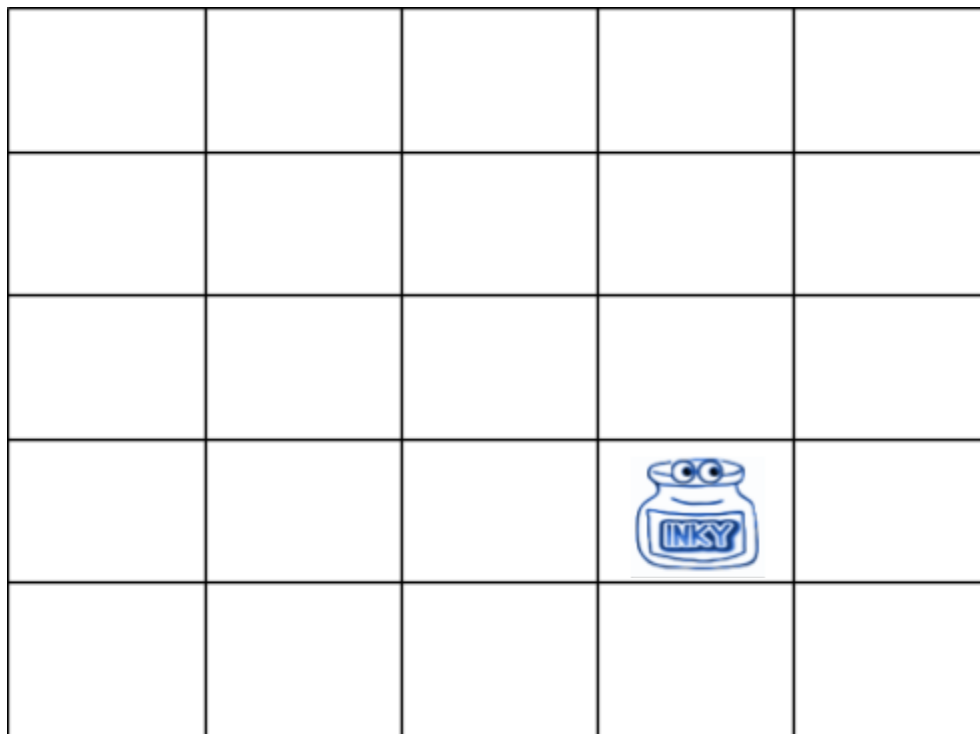
Loop 1 =

3. Circle the pattern.

Next, write directions for Loop 1.

Finally, complete the algorithm.

What did you make?



X → ↑ X ↑ ← X ↑ ← X ↓ ↓
 ↓ ↓ X ↑ ← X ↑ ← X ↑ → X

Loop 1 =

Topic: Computational Thinking**Lesson Title:** Composing a Detailed Algorithm

Lesson Title	Composing a Detailed Algorithm (Lesson 6 out of 6)
Content Standards	<p>CSTA Standards</p> <p>1A-AP-08</p> <p>Model daily processes by creating and following algorithms (sets of step-by-step instructions) to complete tasks.</p> <p>1A-AP-14</p> <p>Debug (identify and fix) errors in an algorithm or program that includes sequences and simple loops.</p>
Content Objective	I can compose an algorithm, using patterns, to achieve a task.
Academic Language	Computational Thinking, Computer Scientist, Algorithm
Sentence Starters	
Assessment	<p>Formal: Teacher will fill out Teacher Observation Form #2.</p> <p>Informal: Students will fill in the Revising Algorithms Packet.</p>
Provisions for Learning Differences	Some students may need assistance.
Materials	Student Interest Form #2 (Post-Assessment)
Learning Activities	
Engage/Instruction (5 mins)	<p>Today you will have an opportunity to make your own algorithm with loops. You will create a challenge of your own that includes a series of steps for others to figure out.</p> <p>For your mission today, you can build an algorithm with AL or Inky. You must use at least one loop in your algorithm. Following completion, your group members will work to figure out your algorithm.</p>

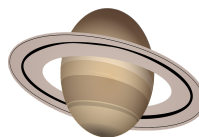
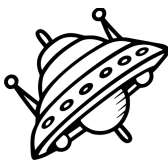
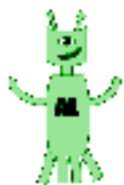
Activity (30 mins)	<p>Students create individual codes using loops. They can write the answer on a separate sheet of paper. If they have more time, they can write an algorithm with a bug in it for a group member to solve.</p> <p>For the second half of the activity, students will present their algorithm to the group. Others can share their thoughts about the algorithm.</p>
Closure (10 mins)	Congratulate students on being computer scientists. Review targets from the unit.

Name _____

My Finest Algorithm-Choice A

Use at least one loop in your algorithm.

Fill in blocks for barriers



Name _____

My Finest Algorithm-Choice B

Use at least one loop in your algorithm.



Name _____

My Finest Algorithm - Answer Key

Use at least one loop in your algorithm.

Loop 1 =

Loop 2 =

Student Interest Form #2

3. Working on computers makes me feel...

4. What does a computer scientist do...

4. Match the term with its definition.

Algorithm	A certain order of events or things
Pattern	A way of completing tasks and solving problems
Computational Thinking	A step-by-step series of instructions to complete a task
Loop	The study of how computers work
Sequence	A series that is repeated
Computer Science	An instruction that represents a series of steps

5. What did you enjoy from this unit of computational thinking?