COMPUTATIONAL THINKING IN ELEMENTARY SCHOOLS: A CASE STUDY

A Capstone Project

Presented to

The Faculty at the Curry School of Education and Human Development

University of Virginia

In Partial Fulfillment

of the Requirements for the Degree

Doctor of Education

---

by

Albert Henry Jacoby, III, B.A., M.A.Ed.

August 2019

Department of Curriculum, Instruction, and Special Education
Curry School of Education and Human Development
University of Virginia
Charlottesville, Virginia

APPROVAL OF THE CAPSTONE PROJECT

This capstone project, “Computational Thinking in Elementary Schools: A Case Study,” has been approved by the Graduate Faculty of the Curry School of Education in partial fulfillment of the requirements for the degree of Doctor of Education.

_____

Co-Chair – Susan L. Mintz, Ph.D.

_____

Co-Chair – Jennifer Chiu, Ph.D.

_____

Committee Member – Joe Garofalo, Ph.D.

May 21, 2019

Executive Summary

Introduction

Computational thinking (CT) is the process of thinking through complex problems logically and is comprised of five core elements: abstraction, generalization, decomposition, algorithms, and debugging (Angeli et al., 2016). Learning to process and break down problems logically, similar to the way that a computer does, encourages students to use technology to become producers of information and content. Students can harness creativity and imagination to solve complex problems through abstraction, pattern generalization, decomposition, iterative thinking, and debugging (Grover & Pea, 2013). Calls to include computational thinking instruction in K-12 schooling promotes equity and access of learning experiences to student groups that otherwise would not be exposed to skills that would widen their economic opportunities later in life. Promoting computational thinking skills better prepares students to be functional members of society capable of solving problems they encounter daily.

Purpose

In order to address the push from global organizations and former President Barack Obama (2016), Virginia legislature mandated that computational thinking be included in the Standards of Learning (SOLs). As a result, the Virginia Department of Education (VDOE) published new SOLs on computer science, which include specific computational thinking skills like abstraction, generalization, decomposition, algorithms and debugging. Classroom teachers in Virginia public schools are expected to teach these new SOLs beginning in the fall of 2019.

The purpose of this capstone study is to understand how elementary school teachers in Rockview County School District (RCSD) make sense of, plan for, and integrate CT into their lessons. Therefore, the problem of practice that I seek to answer in this capstone study is as follows: How can we help teachers implement new standards for computational thinking and infuse the instruction into their lessons?

Methodology

The structure of this capstone study was an embedded, single-case study of four third grade teachers at three elementary schools in one school district who attempted to integrate a new topic into their curriculum. The data collection for this study occurred over an eleven-week period from November 2018 through February 2019. The procedures for data collection included observations of two lessons by each teacher, follow-up interviews after each lesson, and the review of documents associated with lesson planning. The data analysis procedures of data condensation, data display, and verification follows the process outlined by Miles and Huberman (1994). The design of this capstone study sought trustworthiness by addressing credibility, transferability, dependability, and confirmability. The study addressed the confidentiality of participants and the research sites by using pseudonyms.

Findings

The findings of this capstone study reflect the practices of these four third grade teachers only, limiting the generalizations to other teachers. The five findings of this single-case study are as follows:

1. Only three of the eight lessons taught by teachers in RCSD as a part of this capstone study contained elements of CT. Each of the three touched upon one or

more element of CT identified by Angeli et al. (2016) to varying degrees and collectively touched upon all five elements.

2. Teachers whose lessons contained elements of CT used direct, didactic instruction in order to integrate CT into their instruction. Teachers focused on helping students understand CT terms and vocabulary by relating the concepts to students' everyday lives.
3. Teachers in RCSD did not have a common, shared understanding of the meaning of CT as defined by the elements identified by Angeli et al. (2016) or otherwise. Teachers suffered from definitional confusion related to CT and struggled to make sense of their own interpretations of CT, even when provided with concrete definitions and relevant examples.
4. The lessons that touched upon elements of CT were taught by two teachers who used CT resources. Resources include lesson plans, articles, graphics, and instructional videos. Two teachers who did not touch upon elements of CT did not use CT resources.
5. One teacher who taught lessons that touched upon elements of CT used district support personnel. Those personnel accessed and modified CT resources and co-planned the lessons with the teacher.

Implications and Recommendations

Based on the implications of the findings, the recommendations to the VDOE and RCSD include ways for those organizations to promote successful integration of computational thinking and the Computer Science Standards of Learning into classroom instruction. The recommendations are as follows:

- **Recommendation One**: VDOE should adopt a clear, operational definition of CT for teachers, include a “Computational Thinking and Coding” section on grade-level standards documents, and include related CS standards on content area standards documents and curriculum blueprints.
- **Recommendation Two**: RCSD should make use of available professional development opportunities sponsored by the VDOE.
- **Recommendation Three:** RCSD should create a Computer Science SOL Leadership group to support integration of CT and CS standards into instruction across the district. This group would provide professional development opportunities and RCSD-specific instructional resources.

DEDICATION

I dedicate this capstone to my sons, Henry and Sam. Even though I did not know you when I started this journey, it was always for you. I love you both with all of my heart. I also dedicate it to my late father, who I miss every day.

ACKNOWLEDGEMENTS

I could not have completed this capstone project without the support of countless people in my life. I would like to thank some of you here.

To my capstone committee, thank you for your patience and support throughout these several years. Even though my path to completion may not have been the most traditional, your mentorship and guidance were never lacking.

Thank you, Dr. Jennifer Chiu, for being with me for so long. From my interview on a snowy President's Day in 2012, to my very first day of classes and now to my defense, I have learned so much from you, and shared many milestones along the way. You provided the conceptual I look forward to working together to support student learning for years to come.

Thank you, Dr. Susan Mintz, for pushing yourself and everyone around you to achieve their best. You were the first professor to kick me out of a class (for being sick!) and opened a new door for me during a confusing time of my career. I am grateful to you for seeing me through to the end. Enjoy your hard-earned retirement.

Thank you, Dr. Joe Garofalo, for your support through so many of my doctoral assessments. It was been a pleasure to learn from you, in class, through feedback on my work, and from your invaluable experience in academics. I wish you the best in retirement as well.

To Dr. Willy Kjellstrom and Dr. John Baran, my incidental cohort, thank you for paving the way to the finish line for me. The perseverance and determination that each of you demonstrated through tremendous obstacles showed me that even though my path

was not clear, it was still there. Your guidance and feedback, especially in the last four years, were essential to my success.

To the many friends, students, and classmates that I met at Curry, thank you for your friendship, feedback and support. The CED Academy, 3450, Survey Group, and Ed Council—my experiences with you in and out of class made these years exciting, rewarding, and fun.

Thank you to my colleagues in Albemarle County Public Schools, past and present. Many thanks, especially, to Dr. Pam Moran and Ira Socol. The countless conversations about computational thinking, instructional support, academic writing and life in general have provided invaluable assistance along the way.

Thank you to the four third grade teachers in Rockview County School District. Without your willingness and cooperation, this capstone project would not have been possible. To Carmen and Sophie, thank you for answering the call and for your patience for so many years. I always knew you were waiting in my corner.

Lastly, thank you to my family for your unwavering love. As the timeline for this project extended each year, you never doubted me. To my wife and best friend, KC, the value of your rigorous questioning and instructional prowess got me through the many challenges of this entire experience. To my sons, Henry and Sam, your love and joy kept me grounded to what is most important in life. To my mother, Libby, for always thinking that I could do anything, thank you for always believing in me. To my in-laws, Charlie and Sue, your support of me and my family. Whenever we have needed you, you have been there. Lastly, to my late grandparents and father, reminders of your love and support sustain me daily.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter One

Computational thinking (CT) is the process of thinking through complex problems logically and is comprised of five core elements: abstraction, generalization, decomposition, algorithms, and debugging (Angeli et al., 2016). CT involves identifying patterns and commonalities from examples (generalizations) and creating solutions designed for broad reuse (abstraction). CT involves breaking down elements of a problem into component parts (decomposition), developing a sequence of instructions to solve a problem (algorithms). CT also involves identifying errors made in attempts to problem solve when a solution is not reached and correcting those errors (debugging). Definitions and examples of these five elements are summarized in Table 1.

While this capstone project only focuses on CT, it is necessary to consider its relationship to the field of computer science (CS). CT has an important role in CS, but CS is not limited to CT. CS is the study of computer hardware and software, including fundamental principles, ways of use, and the impact computing has on society and the world. CS includes programming, coding, hardware and software development, the internet, and networking. CT is a subset of core concepts and practices within CS that does not include specific hardware or software programs, understanding how the internet works, or networking. CT practices involve creating generalizable solutions to particular problems that computers can instantiate, which has applications to many other domains outside of CS. For example, Example 5 in Table 1 describes a scenario in which someone is using CT to debug a message on an LCD screen connected to an Arduino

microcontroller. Completing the tasks involves manipulation of hardware and software (CS) as well as developing a simple program to achieve the desired solution (CT). Programming and coding are elements of CS that require CT, while successfully assembling the parts of the Arduino and operating the software are not necessarily CT. As this example illustrates, CT plays an important role in aspects of CS.

Table 1

Elements of computational thinking with examples

Element	Definition	Example
1. Abstraction	The skill to decide what information about an entity/object to keep and what to ignore (Wing, 2011).	Identifying the characteristics of five different rectangles that make them all rectangles but ignoring the ways that the rectangles are different.
2. Generalization	The skill to formulate a solution in generic terms so that it can be applied to different problems (Selby, 2014).	Recognizing patterns of "Red-Blue-Red-Red" and "Circle-Square-Circle-Circle" both as ABAA patterns.
3. Decomposition	The skill to break a complex problem into smaller parts that are easier to understand and solve (National Research Council, 2010; Wing, 2011).	Using FOIL (first, outer, inner, last) to solve the following algebraic expression: *(3+4)(8-2).*
4. Algorithms	The skill to devise a step-by-step set of operations/actions of how to go about solving a problem (Selby, 2014).	Writing instructions for an alien to make a peanut butter and jelly sandwich correctly.
5. Debugging	The skill to identify, remove, and fill errors (Selby, 2014).	Programming an Arduino to display "Happy Birthday" on an LCD screen

Note. Adapted from "A K-6 Computational Thinking Curriculum Framework: Implications for Teacher Knowledge," by C. Angeli, J. Voogt, A. Fluck, M. Webb, M. Cox, J. Malyn-Smith, and J. Zagami, 2016, Educational Technology & Society, 19 (3), pg. 50. Copyright 2016 by Journal of Educational Technology & Society.

Learning to process and break down problems logically, similar to the way that a computer does, encourages students to use technology to become producers of information and content. Students can harness creativity and imagination to solve complex problems through abstraction, pattern generalization, decomposition, iterative thinking, and debugging (Grover & Pea, 2013). For example, a student could identify the problem of feeding animals in their science class over the weekend. Students can use decomposition to break down the larger problem into parts, such as knowing when to provide food and the action of providing food. The student can think about patterns throughout the week and abstract and generalize a solution. The student can engage in algorithmic thinking by coming up with a program that knows when to feed the animal based on those generalizations. The student can debug their solution by iteratively testing it over weekends. CT enables students to produce solutions to their problems creative and logical ways.

CT skills can be used in many content domains across K-12 education. Throughout the country, CT is explicitly referenced in state educational standards for science (K12cs.org, 2016). The National Science Teachers Association (NSTA) references computational thinking as a fundamental tool for representing physical variables and their relationships within science and engineering practices (NSTA, 2013). The Next Generation Science Standards (NGSS) outline using computational thinking as one of six science and engineering practices (NGSS Lead States, 2013). The International Society for Technology in Education includes CT in its student standards, as shown in Figure 1 (ISTE, 2016). The Virginia Department of Education (VDOE) included CT in its K-12 Standards of Learning (SOLs) for Computer Science, including ways that those

standards relate to other standards in math, science, English, and social studies (Saunders, 2017). Figure 2 shows how the computer science standards for third graders in the data analysis strand indicate that they are related to math, history and social sciences, and science. These organizations all share the idea that CT is important for all students to learn as other disciplines rely on CT practices. Figure 3 illustrates the relationship between CT and CS, as well as CT to other disciplines.

5. Computational Thinker

Students develop and employ strategies for understanding and solving problems in ways that leverage the power of technological methods to develop and test solutions. Students:

a. formulate problem definitions suited for technology-assisted methods such as data analysis, abstract models and algorithmic thinking in exploring and finding solutions.
b. collect data or identify relevant data sets, use digital tools to analyze them, and represent data in various ways to facilitate problem-solving and decision-making.
c. break problems into component parts, extract key information, and develop descriptive models to understand complex systems or facilitate problem-solving.
d. understand how automation works and use algorithmic thinking to develop a sequence of steps to create and test automated

Figure 1: Computational thinking in the 2016 ISTE Standards for Students (ISTE, 2016, p. 5).

Data and Analysis

3.12 The student will answer questions by using a computer to observe data in order for the student to draw conclusions and make predictions. [Related SOL: Math 3.15, HSS 3.1d]

3.13 The student will create an artifact using computing systems to model the attributes and behaviors associated with a concept (e.g., day and night, animal life cycles, plant life cycles). [Related SOL areas – Math: Models, Science: Moon Phases]

Figure 2: 2017 Computer Science Standards of Learning, Grade Three (VDOE, 2017).

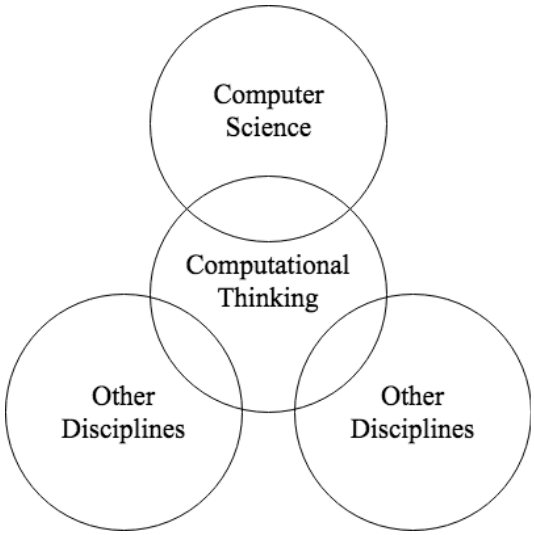


Figure 3: Relationship between CT and CS, as well as CT and other disciplines.

Learning to solve complex problems logically and sequentially can help students learn world languages, sciences, mathematics, history, and social studies. For science and engineering, CT can be thought of as a third pillar alongside theory and experiment to develop ways to understand the natural environment and create novel solutions and technologies. In social studies, CT can help students use rich datasets to find patterns and conduct inquiry in historical contexts. CT can also be used to help students understand the language arts by creating alternative representations through digital storytelling, serve as a basis for writing, or help students conduct textual analysis. CT increases one's "confidence in dealing with complexity, persistence in working with difficult problems, tolerance for ambiguity, the ability to deal with open ended problems, and the ability to communicate and work with others to achieve a common goal or solution" (ISTE, 2011). Teaching students to think computationally across all content areas in K-12 education will make them more logical thinkers and provide them with workforce skills needed for the future as well (Obama, 2016). Real world problem solving requires individuals to generate new representations and patterns to approach the unknown (Lester, 2013). When

students understand how to approach problems with unknown solutions, they may be more confident when facing such uncertainty (M. E. Martinez, 2006; Voskoglou Michael Gr.; Buckley, Voskoglou, & Buckley, 2012).

Problems and Problem Solving

When considering CT as a way of approaching problems to arrive at a solution, it is important to identify what constitutes a problem. Research on problem solving has agreement across all traditions that a problem is a goal which an individual does not know how to reach immediately, and that problem solving is the activity one engages in to achieve that goal (Lester, 2013).

There may be some confusion created by phrases like “word problems” or “story problems”, exercises that simply require mindless computation to solve (NCTM, 2010). This capstone study disregards these problems solved only by routines and calculations, which I consider to be exercises. An example of a non-problem, or exercise, would be asking students to complete a set of mathematical expressions they must solve using the FOIL technique. Students have been taught the FOIL technique in class and will practice using the steps they have learned to solve each exercise. They know what to do in order to get the answer, and in doing so repeatedly, they will develop an increased familiarity to the process. But solving these exercises does not engage students in problem solving.

Problem solving across all domains can benefit from the use of CT. Mathematical problem solving uses CT when students design algorithms to define variables, or scientific problem solving in which geneticists identify similarities between DNA patterns to make gene predictions. Problem solving in social sciences uses CT when urban planners predict demographic shifts based off of patterns in housing development

construction. Therefore, I am open to the exploration of all types of problem solving in this capstone, so long as the complexity of the problems, or tasks, require cognitive engagement and the use of some knowledge and skills, some of which are not routine (Lester, 2013).

Mathematicians like George Pólya and Alan Schoenfeld use problem solving in a way that is similar to CT. Pólya's (1957) four basic principles to problems solving are to understand the problem, devise a plan, carry out the plan, and to look back. Similarly, Schoenfeld (1985) identified problem-solving strategies, or heuristics, as an essential element in determining the quality and success of problem-solving attempts. Schoenfeld named specific strategies in mathematical problem-solving, such as drawing figures, exploiting related problems, reformulating problems, and testing and verification of results. When comparing the elements of CT identified by Angeli et al. (2016) to Pólya's understanding of the problem and Schoenfeld's drawing figures, exploiting related problems and reformulating problems might align with abstraction, generalization, and decomposition. Creating algorithms in CT may be similar to devising and carrying out a plan. Debugging resembles looking back and verifying results. Like CT, the strategies from Pólya and Schoenfeld can also be applied to all domains to improve general problem-solving abilities.

Computational Thinking in K-12 Settings

Emphasizing CT in K-12 settings is not new. Seymour Papert (1980) worked to expand children's CT skills through manipulation of digital objects called "turtles" with the LOGO programming language. LOGO promoted computational thinking, requiring the users to break down goals for their turtles into small tasks that could be debugged

when the turtles failed to perform as intended. Using LOGO and the turtles, the children were able to accomplish challenging tasks, such as how to create different complex geometric shapes efficiently through the use of decomposition, pattern generalization, and algorithms.

More recently, Jeannette Wing (2006) revisited the need for practice in CT in modern education as schools prepare students for their educational and occupational success. Wing's remarks served as a catalyst for researchers in CS to pursue CT learning for all students, not just those studying CS, and her opinion is often cited as a justification for CT for all learners.

CT and other skills related to CS, such as programming and coding, may prove to be useful as students prepare to enter the workforce after they leave school. Projections from the Bureau of Labor Statistics (2019) show the computer and information technology field to be one of the fastest growing job markets at a projected rate of 13 percent from 2016 to 2026. Computer programmers, information security analysts, and software developers are examples of jobs projected to see rapid growth. Engagement in CT in K-12 schools may better prepare students for 21st century careers (Grover & Pea, 2013; Wing, 2006) if projections from the Bureau of Labor Statistics turn out to be true.

Additionally, public school education is intended to prepare students to be functional, contributing members of society. Teachers and administrators model behavior of working together to solve problems they encounter daily. CT can prepare students to make informed decisions and solve problems facing society, such as lack of clean water, diminishing natural resources, or global warming, by giving them the tools they need to simplify problems into more easily managed parts and identify solutions logically. Being

adept at CT skills can help prepare students to benefit society by creating technologies and solutions to pressing global issues.

Conceptual Framework

The elements of CT identified by Angeli et al. (2016) – abstraction, generalization, decomposition, algorithms, and debugging – serve as the conceptual framework for this capstone study (see Table 1, p. 2). It is through these five elements that teacher integration of CT into content-area instruction is analyzed. In the same way that students need CT skills to become functional problem solvers and further contribute to the world around them, teachers need assistance with integrating CT into their curriculum to adequately prepare students for their futures. Very few teachers, if any, are trained on how to integrate CT into their classroom instruction through teacher-education programs (Yadav, Stephenson, & Hong, 2017). Because the international push to incorporate CT into K-12 classrooms is relatively new, I found little evidence of in-service professional development (PD) on CT integration done on large scales for practicing teachers. It will be difficult for teachers to integrate CT into their curriculum if they do not know what it is and have not learned how to teach it. This capstone study seeks to understand how to support teachers integrating CT into their practice. This support can come in the form of instructional resources, support personnel, or PD, and may already be in place in their school or district.

Classroom teachers are expected to be experts in content area curriculum, but there have been no expectations in the past that teachers, particularly teachers who do not teach computer science, be familiar with CT or the elements of CT. That is changing with national and state standards that include CT. But asking teachers to continue to teach

science, social studies, math and language arts, and to also include new instruction of abstraction, generalization, decomposition, algorithms, and debugging creates challenges for teachers. For example, teachers must understand abstraction and how to teach abstraction if they are to engage students in abstraction during lessons on social sciences or mathematics. Additionally, teachers must be able to recognize instances of abstraction that may already be within the curriculum in order understand how the process already plays a role in their current instructional practices. Supporting teachers as they integrate CT serves as the problem of practice for this capstone study. How can we help teachers implement new standards for CT and infuse the instruction into their curriculum, or how can we improve their practices already in place?

With the recent international push to integrate CT into content area curriculum, a number of organizations have created PD and instructional resources for teachers to improve their content knowledge of CT and provide curriculum and lesson plans for teacher use with students. In the research that follows, a PD resource is defined as one that seeks to increase teachers' understanding of the content and the best ways to teach it while an instructional resource is defined as one that is meant to be used with the students, such as a lesson plan or learning activity. Google, The International Society for Technology in Education (ISTE) and the Computer Science Teachers Association (CSTA) provide resources for teaching abstraction, generalization, decomposition, and algorithms (ISTE & CSTA, 2011b). Researchers at Harvard University provide lesson plans that teach algorithms and debugging (Brennan, Balch, & Chung, 2011). Australian researchers developed activities that teach decomposition, algorithms and debugging, and the non-profit Code.org created activities that teach algorithms, debugging, and

generalization (Bell, Witten, Fellows, Adams, & McKenzie, 2015). Resources such as these and others are both PD and instructional, as they can help teachers by increasing their understanding of CT and exposure to different ways to teach CT. In order to address the problem of practice that asks how CT can be integrated into content area curriculum, this capstone study reviewed these resources, focusing on how they target ways to teach the five elements of CT and the scope and quality of curriculum materials provided. The capstone study also investigated if and how these CT instructional resources are used by teachers who are planning to integrate CT into a unit of instruction.

Other forms of PD resource support are district and school-level support personnel. District-level support can come in the form of a number of different specialists who work across a district and at individual schools. For example, a teacher can bring in a district-level Instructional Technology Resource Teacher (ITRT) to assist them with planning or teaching a lesson. School support can also be administration or other teachers within the building. An example of a school-level instructional support could be teachers co-planning lesson plans or discussing past approaches that have been successful during planning meetings. These support personnel may be useful to teachers as they integrate CT into their curriculum, but their own understanding of CT, or lack thereof, could prove to be detrimental to teachers.

The conceptual framework for this capstone study looked at the relationships between teacher thinking of CT, lesson implementation, district and school support personnel, and instructional resources. The framework focused on the ways in which they interact with the five elements of CT: abstraction, generalization, decomposition, algorithms, and debugging.

The problem of practice that I sought to address through this capstone study was: How can we help teachers implement new standards for computational thinking and infuse the instruction into their lessons?

Contextual Background

At a small ceremony at a public military school in Richmond, Virginia on May 16, 2016, a robot delivered House Bill 831 to Virginia Governor Terry McAuliffe to sign (Catrow, 2016; Heiten, 2016). In his ratification of House Bill 831, McAuliffe made Virginia one of the first states mandating all K-12 students learn computer science and computational thinking by including educational requirements for each in the Virginia SOLs. House Bill 831 modified the Code of Virginia § 22.1-253.13:1, Standard 1., instructional programs supporting the Standards of Learning and other educational objectives. The revised Code of Virginia, which became law on July 1, 2016, reads as follows:

> The Board [of Education] shall seek to ensure that the Standards of Learning are consistent with high-quality foundational educational program. The Standards of Learning shall include, but not be limited to, the basic skills of communication, (listening, speaking, reading, and writing); computation and critical reasoning, including problem solving and decision making; proficiency in the use of computers and related technology; computer science and computational thinking, including computer coding; and the skills to manage personal finances and to make sound financial decisions. (Virginia General Assembly, 2016)

The key change in the language of the Code of Virginia is the addition of “computer science and computational thinking, including computer coding.” The Code of Virginia

goes on to say that local school boards are responsible for the development and implementation of instructional programs that align to all of the SOLs (http://law.lis.virginia.gov).

The result of the change in the Code of Virginia was the creation of new CS Standards of Learning (SOL) by the VDOE (Saunders, 2017). The standards, adopted by the VDOE in November 2017, are designed to be integrated within instruction in multiple subject areas from kindergarten through eighth grade, including English, science, history, mathematics, fine arts, and career and technology courses. Standards at the secondary level are separated into independent courses and modules that connect to other content areas where appropriate. Although there is no applicable SOL test associated with CS instruction, it is expected that teachers provide instruction based on these new standards. This requires a change in practice for classroom teachers who are now required to teach additional subject areas, with no additional time and no content removed from their course load by the change in the Code of Virginia.

With the new Virginia CS SOLs calling for the inclusion of CT, general education elementary and middle school teachers will be challenged with integrating CT into their practice. Many teachers may need support to develop needed CT content knowledge and pedagogical expertise for how to teach CT. Moreover, teachers may need help to find ways to connect and integrate CT into other disciplines, as no standards have been removed and instructional time has not changed. As a first step, this capstone study sought to understand how elementary teachers integrate CT into their classrooms.

Rockview County School District

Rockview County School District (RCSD) serves approximately 14,000 students from a large county in Virginia. The district's schools spread across 726 square miles of urban, suburban, and rural areas. Participants in this capstone study worked at three of the fifteen elementary schools in RCSD: Rusty Falls Elementary School, Hilltop Elementary School, and Ridge Elementary School.

The RCSD Department of Instructional Technology (DInT) and Department of Instruction (DI) are exploring various pilot programs that integrate CT at elementary and middle schools across the district due to the new educational requirements from the VDOE. Four third grade teachers were observed while teaching lessons that integrate the new standards with a focus on CT as a part of this capstone study.

RCSD administrators have concerns about the new state requirements, and expect that teachers need support to integrate CT into their practice (personal communication, June 1, 2016). In a survey of preservice elementary school teachers enrolled at a university near RCSD, none of the 29 students surveyed were majoring mathematics, CS, or engineering, fields related to the new standards (Kjellstrom, 2017). These numbers are representative of the greater school district. Anne Butters, Teacher Licensure Specialist for RCSD, confirmed that there are no elementary teachers employed in the district who hold endorsements in a field related to the new standards in their employment records (personal communication, August 9, 2017). As teachers in RCSD prepare to teach the new standards, their instructional strategies are of paramount importance to their supervisors. School- and district-level administrators hope to use information from pilot

programs to develop plans for support of all teachers across the district who will teach the new standards.

Research Questions

In order to address the problems of practice for this capstone project, I sought to answer the following research questions, derived from Grossman's (1989) construct of PCK and research on CT:

1) How do elementary teachers in Rockview County School District integrate computational thinking into their instructional lessons?
    a) What elements of computational thinking are present in the lessons?
    b) To what extent is computational thinking effectively integrated into the instructional lessons?
    c) What activities or tasks are students engaged in that purportedly teach the elements of computational thinking?
    d) What types of instructional strategies are teachers using to integrate computational thinking into their instructional lessons?
2) How do elementary teachers in Rockview County School District define computational thinking?
3) How do elementary teachers in Rockview County School District prepare to integrate computational thinking into their instructional lessons?
    a) To what extent do teachers use available computational thinking resources when integrating computational thinking into their instructional lessons?

b) To what extent do teachers use district and school support personnel to plan for instruction that integrates computational thinking into their instructional lessons?

Overview of Methods

This capstone study was a case study with an embedded, single-case design (Yin, 2014) that analyzed four third grade classroom teachers, in three school contexts, in RCSD. Yin (2014) argues that case study research methods are preferred when a "how" question is being asked about a contemporary set of events over which the researcher has little control. The units of analysis within the school contexts were the classroom teachers, as teachers are primarily responsible for providing instruction to students in the classroom setting. Data collection included audio recordings and observations, through the use of an observation protocol, of four teachers during instruction of lessons in two different content areas chosen by the teachers that integrated CT, audio recorded interviews with teacher participants, and teacher-provided documents related to CT integration, such as e-mail exchanges with school and district support, lesson plans, classroom activities, formative and summative assessment, and projects. This data provided insight into the problem of practice which focuses on how to support elementary school teachers to integrate CT into instruction.

Case study participants included four third grade teachers in RCSD. Data collection occurred during the second and third nine-week grading periods of the 2018-2019 school year; audio recordings and teacher observations occurred during instruction of a total of two lessons by each teacher that purportedly integrated CT into two different content areas. The lessons and content areas were selected by the participants. Data

triangulation from the observations, interviews, and collected documents created and maintained validity and credibility throughout data collection and analysis. Table 2 outlines the relationship between the research questions and how the information helped to address the problems of practice.

Table 2

What I need to know and why I need to know it

What do I need to know	Why do I need to know this
What CT knowledge do teachers demonstrate in their teaching and planning?	To identify their knowledge about CT and determine ways to support development of that knowledge
What CT instructional practices do teachers use to teach?	To identify their knowledge about teaching CT and identify ways to support CT instructional practices
To what extent and how do teachers use instructional resources as they teach CT?	To examine the instructional resources used by teachers as they integrate new content and identify characteristics of successful resources to improve instruction of CT
To what extent do teachers use district and school support personnel as they teach CT?	To examine the support district and school support personnel used by teachers as they integrate new content and look for ways to improve support to improve teaching of CT

Data collection and analysis occurred concurrently and followed the methods outlined by Miles and Huberman (1994) through three concurrent processes: data condensation, data display, and conclusion drawing/display. Chapter Three contains a complete explanation of research methods for this capstone study, including expanded information about the research site, participants and researcher's role. In chapter four of this capstone study, I address the problems of practice through presentation of analysis of

the teachers' content knowledge, pedagogical content knowledge, and integration of CT into instructional lessons. I looked for patterns in use of supports to develop conjectures about what kinds of instructional resources and supports helped teachers integrate CT into their lesson.

Definition of Terms

The following is a list of definitions of key terms the way in which they apply to this capstone study:

- Computational thinking (CT) is the process of thinking through problems logically, breaking them down into smaller parts, and organizing those parts to solve the problem correctly and efficiently, in such a way that that solving the problem can be executed by a computer.
- A problem is a goal where the means to achieve the goal are not immediately known.
- Computer Science (CS) is the study of computer hardware and software including fundamental principles, ways of use, and the impact computing has on society and the world that encompasses computational thinking.
- Coding is the act or ability of using a computer programming language.
- Programming is the process of writing in a language that can be processed and enacted by a computer.
- Algorithms are expressions used to explain solutions to mathematical or computational problems.
- Generalization is recognizing patterns or common features among specific examples or instances of a problem.

- Abstraction is using generalizations to create reusable programs or solutions that reduce complexity of a problem.
- Decomposition involves breaking down larger problems into smaller constituent parts.
- Iterative thinking is a process involving design of a solution, implementation of the solution, and review of the results.
- Debugging is identifying and fixing errors in programs or algorithms.

Summary

This introductory chapter provided background on CT, contextual information related to the identified problem of practice and provided an overview of the capstone project. It also described the conceptual framework that looks at the ways that teachers plan for and teach CT and the elements of CT. Chapter two examines the literature on CT, the importance of CT, research on teaching CT, and resources available to teach CT in elementary school classrooms. Chapter three outlines the methodology for this capstone project, including data collection procedures and data analysis methods. Chapters four and five details the findings of the study, implications, and the recommendations. Chapter six is an action communication about the findings, implications, and recommendations written to RCSD and the VDOE.

Chapter Two

Increased attention to careers in computing and pressure from national organizations has led to state-mandated standards in computational thinking (CT) in Virginia. CT is the process of thinking through problems logically, breaking down elements of the problem into component parts, and organizing and approaching those parts in ways that make it easier to solve the problem. The purpose of this capstone study is to address problems of practice focusing on the integration of CT into content area instruction. This problem of practice is specifically focused on ways that we can help teachers to integrate CT and the supports that are available for teachers to do so. This chapter focuses around four questions that were derived from this problem of practice and the research questions of this capstone study, focusing on teacher PCK:

- What is computational thinking?
- Why should computational thinking be taught?
- What do teachers need to know in order to teach computational thinking?
- What resources are available to help teach computational thinking?

CT is relevant in many domains of study, but it is particularly necessary to the domain of computer science (CS). Although this capstone study is interested in integration of CT not necessarily related to CS, this literature review begins with background information on CS and its relationship with CT. CT has an important role in CS, but CT is not limited to CS. While this capstone study attempts to solve a problem of

practice of teachers' integration of CT into content-area lessons, the literature review will reference some resources created specifically for teaching CT through the lens of CS as well as those other content-area domains. It is appropriate to view CT within the context of these CS resources because of the relationship between CT and CS, as well as the inclusion of CT in many organizations' frameworks or standards for CS. Limiting the scope of this literature review to CT found outside of the domain of CS would be excluding valuable resources from review.

Early research in CS education

Efforts in CS education have progressed in the last two centuries. In the early 19th century, Lady Ava Lovelace, the "mother of computer science," annotated a presentation on Charles Babbage's "Analytical Engine" (Coe & Ferworn, 2016). Lady Lovelace's notes paved the way for Alan Turing (1950), the "father of computer science," to speculate about future advances in engineering and programming that would allow computers to complete limitless tasks—perhaps even to imitate the logical thinking and reasoning abilities of humans. However, it was Turing's contemporary, John von Neumann, who first used the term "computer science" in his writings on the computational abilities of early computers (Denning, 2017; Szász, 2011).

Since von Newmann and Turing's time, CS has made gains in its legitimacy as a domain of study (Denning, 2009; Denning et al., 1989; Knuth, 1972). The Computer Science Teachers Association (CSTA) defines CS as "the study of computers and algorithmic process, including their principles, their hardware and software designs, their applications, and their impact on society" (Seehorn et al., 2011, p. 1). CT, a way of thinking about complex problems in such a way as that they can be solved by a computer,

is a necessary skill to the principles of CS, particularly to the algorithmic process. In fact, the difference between CT and algorithmic thinking in the literature is unclear (Buitrago Flórez et al., 2017), and Denning (2009) states that the two are synonymous.

Attempts by computer scientists to enhance children's CT skills were popularized in Seymour Paper's 1980 book *Mindstorms: Children, computers and powerful ideas*, through the use of CS exercises. Denning (2017) credits Papert with the first use of the term *computational thinking*. Papert developed a computer programming language called LOGO, where the user manipulated a digital "turtle" by writing procedural codes. Through his research at MIT with LOGO, Papert taught children how to break down problems into smaller components and "debug," or problem-solve, when the turtles didn't behave as expected. In the nearly four decades since *Mindstorms*' publication, enhancements in computer processing and graphic ability have allowed for a resurgence in talk amongst CS educators focusing on children's CT, hoping to build on Papert's work (Voogt, Fisser, Good, Mishra, & Yadav, 2015).

Twenty-six years after Papert introduced the world to the term "computational thinking," Wing (2006) penned a seminal viewpoint calling for CT to be taught to everyone (ISTE & CSTA, 2011a; Israel, Pearson, Tapia, Wherfel, & Reese, 2015; Seehorn et al., 2011) based on her own high valuation of CT. Wing's piece for the Communications of the ACM has served as a stepping stone for others advocating for CT education, as it has been cited over 3,000 times since publication twelve years ago (scholar.google.com). Wing compared CT to ubiquitous computing, the latter being "yesterday's dream that became today's reality; computational thinking is tomorrow's

reality" (p. 34). Wing provides over two dozen examples of what CT can be, such as recursive thinking, abstraction, decomposition, and heuristic reasoning.

What is Computational Thinking?

Wing failed to provide an explicit definition of CT in 2006, and since then many groups have worked to develop a working definition of CT, including the National Academy of Sciences (National Research Council, 2010), Furber (2012) for the Royal Society, the Computer Science Teachers Association (CSTA), and the International Society of Technology in Education (ISTE) (Angeli et al., 2016). As a result, CT suffers from definitional confusion, with each group championing their own slightly different version of the same concept. This confusion makes identifying instances of CT challenging.

This capstone study adopts the definition developed by Aho. Aho (2011) defined CT as "the thought process involved in formulating problems so their solutions can be represented as computational steps and algorithms" (p. 2). Since then, Aho's definition has been referenced or slightly modified by Angeli et al. (2016), Denning (2017), K12CS.org (2016), and the Virginia Department of Education (VDOE) (Saunders, 2017). This capstone study project adopts Aho's definition due to the fact that it is referenced by the VDOE in their CS Standards of Learning (SOL). It is important to have a consistent definition in this capstone study and also when implementing new content standards so as to decrease confusion and ambiguity amongst the case study participant teachers about the intended topic of instruction.

CT is a way of thinking about a complex problem so that its solution can be identified logically and sequentially. It can also be defined by five elemental components

identified by Angeli et al. (2016) and defined in Table 1: abstraction, generalization, decomposition, algorithms, and debugging. Understanding of these elements can further explain the purpose and value of computational thinking. To abstract is to identify and ignore extraneous information about a problem and focus on what it is important to reduce complexity and create reusable solutions. To generalize is to identify patterns and characteristics out of specific examples or instances. Decomposition is breaking down a complex problem into smaller parts that are easier to understand and solve. Algorithms refers to the ability to organize a solution into a set of operations that can be followed sequentially. Lastly, debugging is the evaluative component of computational thinking, where one assesses whether the proposed solution works and if not, identifies and corrects persisting errors. Breaking down computational thinking into its elements can simplify its definition as well as promote applicability to all domains of study. These five elements of CT serve as the basis for the conceptual framework of this capstone study.

It is important to further clarify the relationship between CT and CS, computer programming, and coding. CS researchers emphasize CT because development of CT skills are essential to the teaching and learning of CS (Buitrago Flórez et al., 2017). CS encompasses hardware, software, and how things like the internet and robotics function. CT involves practices and concepts that bring together critical elements of CS, namely problem solving and algorithmic thinking, which can be applied to other disciplines. Computer programming is the process of writing in a language that can be processed and enacted by a computer. Programming represents a physical manifestation of the elements of CT. Coding is the act or ability of using a computer programming language. Thus, CS represents the entire domain made up of things like hardware, software, and networking,

CT involves critical practices and concepts within the domain, and programming involves using specific tools to code solutions in CS.

Computational Thinking and Problem Solving

By defining CT as a certain way to go about solving a problem, is it important to clarify what is meant by a problem. Research on problem solving across all traditions agrees that a problem is "a task for which an individual does not know (*immediately*) [emphasis added] what to do to get an answer" (Lester, 2013, p. 247). I emphasize *immediately* because that creates the distinction between a problem and what I will refer to as an exercise. An exercise is a task for which an individual immediately knows what to do to get an answer. For example, the multiplication problem 12 x 3 is an exercise solved regularly by elementary students who have learned multiplication facts or ways of completing two-digit by one-digit multiplication. It may take them time to complete the exercise and arrive at the correct solution, but they know *how* to solve it immediately.

In contrast, the example provided in Table 1 of someone programming an Arduino to display a message meets the qualifications of a problem. The user must learn to program the Arduino and write the correct codes, in order to display "Happy Birthday." If an error occurs and the message fails to display, the user will debug the code in an order to identify and correct the error. After learning to program the Arduino to display a message and developing the ability to apply this knowledge to similar situations, the solution to a similar task (such as displaying "Hello") will become immediately known to the programmer. When faced with a similar task, it will merely be an exercise, rather than a problem. Solving the initial problem required the programmer to engage in a variety of cognitive actions, namely decomposing the problem, creating

algorithms, debugging any issues, and generalizing the actions to be applied elsewhere. These actions required some knowledge or skill, some of which may not have been routine. Engagement in these non-routine cognitive actions allowed the programmer to solve the problem. CT skills can help people to complete tasks with solutions that are not immediately known.

Why Should Computational Thinking be Taught?

Wing (2006) was the first to claim the value and universal importance of CT for all. She argued that the ability to think computationally is as fundamental a skill as reading, writing, and arithmetic, and that it is necessary for success in a variety of content areas (Buitrago Flórez et al., 2017; Wing, 2006; Yadav, Good, Voogt, & Fisser, 2017). CT can have a direct connection to a number of fields, such as biology with metabolic pathway reconstruction, digital synthesization in music, and medical surgery automation in medical practices. Table 3 shows more fields in which CT can be directly applied with examples of applications. Angeli et al. (2016) propose that teaching CT as a basic skill across the school curriculum will enhance K-12 students' abilities to use abstraction, algorithmic and logical thinking, and skills at problem solving. Lye & Koh (2014) argue that aptitude for 21st century skills like creativity, critical thinking and problem solving, can all traced back to understanding CT.

In addition to developing 21st century, higher-level thinking abilities, being able to think computationally can better prepare students for the workforce they will enter when they conclude their formal education (Fluck et al., 2016). According to a 2017 report from the Bureau of Labor Statistics (https://www.bls.gov/ooh/computer-and-information-technology/home.htm), careers in computing related fields are projected to

be one of the fastest growing and highest paying job markets in the United States, with an expected 13 percent increase in new jobs from 2016 to 2026. Interested students need to be prepared to enter this expanding job market, and early exposure to CT may promote an interest in computing (Armoni & Gal-ezer, 2014; Furber, 2012; Israel et al., 2015; Nelson, Sahami, & Wilson, 2016; Ouahbi, Kaddari, Darhmaoui, Elachqar, & Lahmine, 2015; Partovi, n.d.; Yadav, Good, et al., 2017; Yadav, Mayfield, Zhou, Hambrusch, & Korb, 2014).

Promoting an interest in computing through CT in K-12 education supports equity and access (K12cs.org, 2016). Exposure to computing has traditionally been limited to white males (Margolis, Estrella, Goode, Holme, & Nao, 2008). According to the National Center for Education Statistics, only 17.6% of undergraduate degrees in computer science were awarded to women in 2010-2011 (U. S. Department of Education, 2012). Since that time, the CS advocacy group Code.org has found that since the organization's founding in 2013, participation by females, underrepresented minorities, and students from low socioeconomic status in Advanced Placement CS courses has increased (https://code.org/diversity). The goal of organizations like Code.org is to provide all students with basic skills that will enable them to be productive members of society and make informed decisions throughout their lives (Saunders, 2017). When CT instruction is accessible to all students diverse in terms of race, gender, socioeconomic status, disability, and English language proficiency, more students will have exposure to knowledge and skills that can help them to be successful in their careers and better equipped to use critical thinking and problem solving in the world around them (K12cs.org, 2016).

Table 3

Fields in which computational thinking can be directly applied

Area	Sample activities related to computational thinking
Biology	Genome sequencing, genome assembly, and gene prediction Protein interaction modeling Metabolic pathway reconstruction Population and systems biology
Chemistry	Discovery and design of drugs Molecular dynamics simulations Chemical pathways modeling Study of fundamental properties of atoms and molecules
Physics	Physical behavior of materials Optical performance simulations Astrophysics modeling Physical interaction in biomolecules
Medicine	High-throughput biomedical assays Neuronal pathways modeling Physiology performance simulations Medical surgery automation Analysis of drugs and environmental toxins Medical illustration
Engineering	Mobile and internet computing Gaming Robotics Computer and network security Artificial intelligence Data base systems
Arts	Image design Movie animation
Music	Acoustics simulations Recording techniques Sound synthesis and manipulation Computer music
Social Sciences	Demographic simulations Pandemic reaction modeling Computational finance

Note. Adapted from "Changing a generation's way of thinking: Teaching computational thinking through programming," by F. Buitrago Flórez, R. Casallas, M. Hernández, A.

Reyes, S. Restrepo, and G. Danies, 2017, Review of Educational Research, 87 (4), pg. 839. Copyright 2017 by American Educational Research Association.

At this time, CT integration is in an emergent state, and there is a dearth of empirical studies supporting the value of CT outside of the field of CS or other computational fields like Bioinformatics, or Computational Statistics. Calls for its universal inclusion are largely ignored or unheard by educators outside of those fields. Supporters of ubiquitous CT education rely on the word of scholars in the field of CS, industry leaders, and even former United States President Barack Obama to support calls for universal instruction of CT in K-12 education. Claims that CT is a valuable tool and will increase problem-solving skills across all domains are unsubstantiated in the research, although they do appeal to common sense. Martinez & Stager (2013) summarized this issue well by pointing out that "government and business leaders value and want schools to teach… [CT and that it has been] deemed necessary for United States economic progress and global competitiveness" (pg. 44). The argument that an introduction to CT at a young age will better prepare students for future jobs in the field seems logical but is largely hypothetical in nature. CT research needs to be more deeply informed in terms of implementation within the K-12 curriculum.

Despite the lack of empirical evidence of the benefit of an education that implements CT through abstraction, generalization, decomposition, algorithms, and debugging, many organizations continue to work towards the goal of all students learning CT. This may be because they see the value of CT education anecdotally, even if it cannot yet be demonstrated empirically at this time. Their continued support for CT education in K-12 schools has led to the development of a number of resources for teaching and learning CT.

What Do Teachers Need to Know in Order to Teach Computational Thinking?

Given the national and state efforts to increase CT instruction in K-12 settings, it is increasingly important to understand how to support teachers to develop the pedagogical expertise needed to integrate CT in their practice. The following section begins with organizational efforts to improve teacher instruction of CT, and then outlines relevant work on improving teacher instruction.

Developing Teacher Understanding of Computational Thinking

In an effort to provide teachers with clarity and understanding about CT and its value in the K-12 curriculum, some organizations have created resources with advocacy information and content knowledge for teachers. K12cs.org offers a free K-12 Computer Science Framework with a section focusing on CT (2016). K12cs.org references CT as a crucial component in four of the seven course principals listed in the K-12 Computer Science Framework, showing CT to be at the heart of CS practices (K12cs.org, 2016). Those practices are recognizing and defining computational problems, developing and using abstractions, creating computational artifacts, and testing and refining computational artifacts. The K-12 Computer Science Framework acknowledges the fact that as K-12 CS education is still in a nascent state, most research has been done by practitioners in the field, if any research was performed at all (K12cs.org, 2016)

Additionally, Computing at School published “Computational thinking: A guide for teachers” (https://community.computingatschool.org.uk/resources/2324). This guide provides teachers with definitions of its concepts of CT, including logical reasoning, abstraction, evaluation, algorithmic thinking, decomposition, and generalization. Computing at School describes five different ways of “computational doing,” where

students can be observed reflecting, coding, designing, analyzing, and applying. The guide also provides a long list of distinct learner behaviors that can be observed to show different elements of CT have been demonstrated. Examples of these behaviors are found in Table 4. Resources such as the K-12 Computer Science Framework and Computing at School let teachers know what about CT should be taught to students and provide teachers with background about CT to support their own understanding. Once teachers develop their own understanding of CT, they must learn to teach CT to others. Studies such as these address different aspects of teaching CT in K-12, but there are no comprehensive studies that reflect empirical research on K-12 CT integration (Ilic, Haseski, & Tugtekin, 2018).

Table 4

Sample observable learner behaviors associated with computational thinking

Element	Behaviors
Abstraction	• Reducing complexity by removing unnecessary detail. • Hiding the full complexity of an artifact (hiding functional complexity).
Generalization	• Identifying patterns and commonalities in artifacts • Adapting solutions, or parts of solutions, so they apply to a whole class of similar problems.
Decomposition	• Breaking down artifacts into constituent parts to make them easier to work with. • Breaking down a problem into simpler version of the same problem that can be solved in the same way (recursive and divide and conquer strategies).
Algorithmic thinking	• Formulating instructions to be followed in a given order (sequence). • Grouping and naming a collection of instructions that do a well-defined task to make a new instruction (subroutines, procedures, functions, methods).
Debugging	• Assessing whether an artifact does the right thing (functional correctness. • Making trade-offs between conflicting demands.

Note. Adapted from "Computational thinking: A guide for teachers," by A. Csizmadia, P. Curzon, M. Dorling, S. Humphreys, T. Ng, C. Selby, and J. Woolard, 2015, pgs. 14-15. Copyright 2015 by Computing at School.

Improving Teacher Instruction

Instructional best practices can be developed through teacher PD, or activities that are intended to improve performance in their current or future roles of working with students and teachers (Little, 1987). PD can take place through discrete activities such as college courses, conferences, and workshops, or through situated learning experiences such as discourse and community practices as a part of formal or informal learning communities, co-teaching, mentoring and reflection, book clubs, or even curriculum

materials review (Desimone, 2009). Buchholz et al. (2013) recognized key factors in these PD opportunities that are necessary for improving teacher understanding of ways to teach: duration, peer debriefing, and opportunities for reflection. They first argue that teacher understanding of ways to teach develops over time as teachers gain more experience. Additionally, teachers must have opportunities to experiment with different methodologies in their own classrooms. Peer debriefing with fellow teachers allows for the exchange of ideas, strategies, and solutions, which can also increase teacher understanding of ways to teach. Lastly, the willingness and opportunity to reflect on one's own teaching practices. When teachers critically reflect in order to make sense of experiences in the classroom, they can gain valuable insight into the difficult practice of teaching. van Driel & Berry (2012) conclude that efforts to develop teacher understanding of ways to teach should mirror their professional practice, providing them with opportunities to innovate instructionally and to reflect on their experiences, individually and collectively, throughout their careers.

van Driel et al. (1998) insist that teachers must be able to exchange ideas, strategies, and solutions with other practitioners to build understanding of ways to teach. As CS teachers often have limited exposure to other teachers in their discipline, Go & Dorn (2016) chose to analyze an exchange of ideas through online communities. Two electronic mailing lists, the CS Teacher's Association (CSTA-MEMBERS) and the CS Teaching Tips (CSTT) provided contrasting styles of electronic discourse that teachers could engage in to develop their understanding of ways to teach CS. CSTA-MEMBERS allows participants to ask questions about anything related to CS, which was both an advantage and a disadvantage of the list-serv. Text analysis revealed that discussion of

ways to teach CS was certainly present within the CSTA-MEMBERS messages, but so were a wide range of other CS education topics, making it difficult for participants to filter different types of information. CSTT, in contrast, is devoted entirely to ways to teach CS. Browsing the CSTT site gives teachers the option to search for the information that they need but does limit them from being able to ask anything about CS teaching practices. The researchers conclude that these resources could be valuable to practicing teachers as they work to expand their understanding of ways to teach CT.

Understanding of Ways to Teach Computational Thinking

The following are examples of work done specifically to identify or improve ways of teaching CT, described in order of least relevant to most relevant to this capstone study project. CT is embedded into the Dutch national curriculum for secondary teachers (Grgurina, Barendsen, Zwaneveld, van Veen, & Stoker, 2014). Tolboom, Krüger, & Grgurina (2014) conducted an online survey of Dutch secondary CT teachers in order to learn about their perceptions, beliefs and hindrances around CT. They found that many teachers "hold alternative beliefs or views which are not in line with the prevailing definitions and nature of Computational Thinking" (Grgurina et al., 2014, p. 173). This is likely due to prevailing definitional confusion caused by multiple organizations each creating their own operating definitions, as described above. This survey may indicate that students will have difficultly learning CT due to disagreement from their teachers about what CT is, even in a country where CT instruction is embedded within their curriculum.

While the secondary teachers in the Netherlands revealed confusion about the definition of CT and ways to teach it, elementary teachers are not required to teach CT,

and thus may be presumed to know even less about teaching CT. To counter this, Angeli et al. (2016) designed a course on how to teach CT to K-6 students. The participants in the course were fifteen elementary school teachers pursuing a master's degree in instructional technology. The course instructor engaged teachers in authentic problem solving, identifying ways to improve people's lives in the cities and towns that the teachers resided, and taught the teachers how to create abstract models without using computer programs. The teachers, who had no prior experience with computer programming, then used Scratch to develop programs based off of their models to solve the real-life problems. CT concepts such as sequencing, loops, and parallel processing were taught explicitly and demonstrated with many sample codes and illustrations. Researchers found that these veteran teacher education students had no problem learning the CT concepts, although variables and conditional logic were most challenging for the group. This study shows that teachers may be able to learn how to teach CT through the use of computer programming and Scratch. By completing programming activities that caused them to engage in computational thinking, teachers saw first-hand ways to introduce their students to CT while simultaneously expanding their own understanding of CT.

What Resources Are Available to Help Teach Computational Thinking?

This capstone study sought to answer a problem of practice in which teachers must integrate CT into content area curriculum. Analyzing the instructional and PD resources used by these teachers was an essential part in making recommendations for improving instructional practices. The analysis of specific resources used by teachers during this capstone study took place during the data collection phase. The section that

follows will review resources found and referenced while researching CT in K-12 education, and specifically elementary education. Thanks to organizations like Code.org and the ACM, there has been a resurgence in the call to teach CT and the creation of resources to do so (Armoni & Gal-ezer, 2014; Kumar, 2014b, 2014a; Prottsman, 2014). All of the resources described below are available online for free and offer curriculum for K-12 education. Because this capstone study focused on the practices of third grade teachers, attention in this review is directed towards resources available for use in the middle elementary grades. Some of the materials created specifically address computational thinking and direct their resources in that manner. Other resources have been created to promote the CS, but still include instructional strategies for the elements of CT. Analysis of these resources focuses on how each addresses CT components of abstraction, generalization, decomposition, algorithms, and debugging. Table 5 summarizes the resources designed for teaching CT, identifies the focus of their materials and lists the elements addressed.

Table 5

Summary of resources available to learn about and teach CT in elementary schools

Resource	Materials focus	CT specific	CT elements
Code.org (http://code.org)	Teacher knowledge curriculum, independent activities for learners	No, general CS	Ab, G, Dc, Al, Db
Exploring Computational Thinking (Google, n.d.-c)	Teacher knowledge, curriculum	Yes	Ab, G, Dc, Al
Computational Thinking for Educators (http://computationalthinkingcourse.withgoogle.com)	Teacher knowledge, curriculum	Yes	Ab, G, Dc, Al
Computational Thinking Teacher Resources (ISTE & CSTA, 2011)	Teacher knowledge, curriculum, lesson plans	Yes	Ab, G, Dc, Al
Computational thinking: A guide for teachers (Csizmadia et al., 2015)	Teacher knowledge	Yes	None
K-12 Computer Science Framework (K12cs.org, 2016)	Advocacy, teacher knowledge	Some	None
Creative Computing (Brennan et al., 2011)	Teacher knowledge, lesson plans	Some, general CS	Al, Db
CS Unplugged: An enrichment and extension programme for primary-aged children (Bell et al., 2015)	Student activities	No, general CS	Dc, Al, Db

Note. Ab = abstraction; G = generalization; Dc = decomposition; Al = algorithms; Db = debugging

Computational Thinking-Specific Resources

Three resources reviewed focus on the learning and teaching of CT and address four elements of CT, specifically abstraction, generalization, decomposition, and algorithms. These resources seek to build teacher content knowledge and provide curriculum for teaching CT to students. Two of the resources are provided by Google, and one is a collaboration from ISTE and CSTA.

Google created a free online resource for teaching CT (Google, n.d.-c). The site includes opportunities for educators to integrate CT into curriculum areas like mathematics, science, English/language arts, music, and art, design and media studies. For example, one resource available for teaching US history to students ages 8-18 focuses on map visualization with a program called Pencil Code. Pencil Code "provides a simple way to illustrate statistics geographically by drawing bubbles on a map. Students can analyze, fill in, or change parts of the program" (Google, n.d.-c). This activity teaches abstraction and generalization. Students must identify and extract relevant information to define main ideas and observe patterns in the data. The code for this activity and a sample output are shown in Figure 4. The full activity resource is found in Appendix A. A lesson plan for music with students ages 11-18 "allows students to examine the various aspects of music such as scales, melody, and rhythm. The patterns they discover will enable them to modify an algorithm to improve the quality of the music generated by the algorithm" (Google, n.d.-c). The full lesson plan is found in Appendix B. These are just two examples of over thirty resources available for teaching students middle elementary school grades. Through this resource, Google provides instructional support for teachers working to integrate CT into their curriculum.

Code

```
Somalia = new Turtle
Somalia.wear 'http://goo.gl/4E8BcV'
Somalia.css 'z-index', -1
moveto 110, 80
dot yellow, 30
label "Eyl"
moveto -25, -105
dot skyblue, 70
label "Merca"
ht()
```

Sample Output

Figure 4: Code and sample output for Pencil Code: Map Visualization

Google also has a self-paced online course for educators called “Computational Thinking for Educators.” The goal of the PD course is to teach educators about CT, how it is different from CS, and how to integrate CT across multiple content area domains. By participating in the course, educators will increase their awareness of CT through exploration and experimentation with CT examples integrated into content areas and developing a plan to integrate CT into their own classrooms ((Google, n.d.-b). Course content includes instructional resource modules in exploring algorithms, finding patterns, and developing algorithms across four content areas: CS, humanities, math, and science. The humanities activity for exploring algorithms exposes the learner to Google’s ngram viewer, which “visually displays the occurrence of a word, or phrase (ngrams) for many of the books Google has scanned” (Google, n.d.-a). Figure 5 shows the ngram viewer for a search of the phrase “computational thinking” from 1950 to 2018. The activity poses questions to the learner about how people might perform tasks completed by algorithms in the ngram viewer. The course concludes with a project, where participants must submit a lesson plan integrating CT. This course does primarily emphasize enhancing teacher

understanding and teaching ability related to CT, but in focusing on the later, it does provide curriculum and resources for teaching CT.

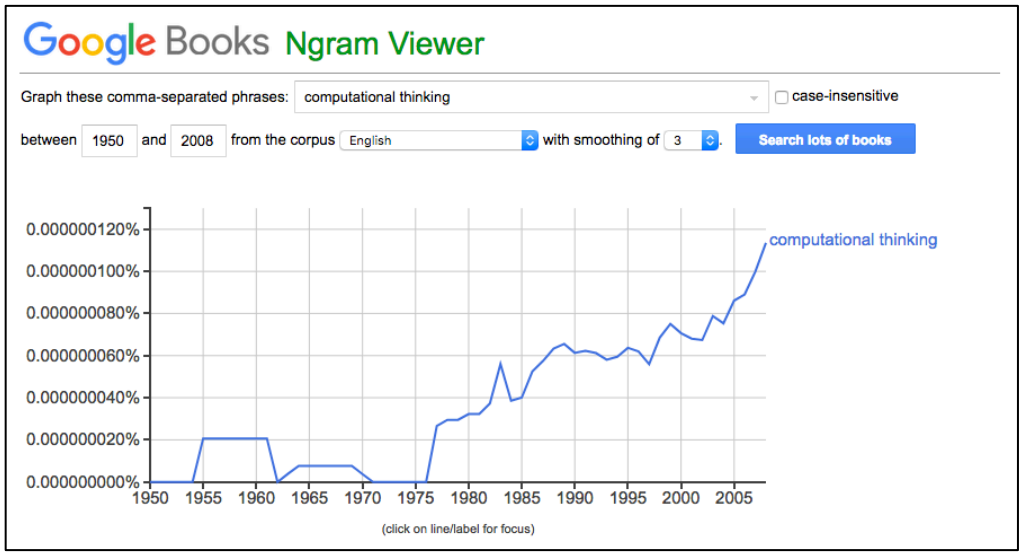


Figure 5: Sample ngram viewer from a search of "computational thinking" from 1950 to 2017 (Google, 2013)

ISTE and CSTA, with funding from the National Science Foundation (NSF) created a "Computational Thinking Teacher Resources" guide (2011b). In addition to PD teacher background information such as an operational definition of CT and a vocabulary and progression chart (see Table 6), the guide provides instructional resources through nine learning experiences for teachers to use to integrate CT activities into their curriculum. Four of the experiences target K-5. Each K-5 experience provides outcomes, standards alignment, evidence, activities, strategies, and resources, as well as a "CT guide on the side" with tips related to CT-related vocabulary. Table 7 summarizes the purposes of the four elementary experiences.

T*able 6*

Sample of computational thinking vocabulary and progression chart

Term	Definition	Grades PK to 2	Grades 3 to 5
Problem Decomposition	Breaking down tasks into smaller, manageable parts	Create directions to a location in the school by breaking the directions down into smaller geographical zones.	Develop a plan to make the school "green." Separate strategies such as recycling, paper and cans, reducing use of electricity, and composting food waste.
Abstraction	Reducing the complexity to define main idea	With many sizes and colors of three-sided shapes, the abstraction is a triangle.	Hear a story, reflect on main items, and determine the appropriate title.

Note. Adapted from "Computational thinking teacher resources", 2nd edition, by International Society for Technology in Education (ISTE) and Computer Science Teachers Association (CSTA), 2011, p. 8. Copyright 2011 by CSTA and ISTE.

Table 7

Summary of elementary computational thinking learning experiences

Experience Content Area Grades	Standards	Outcomes	CT skills/dispositions
1. Sequencing Language Arts PK-2	Grade 2 Common Core English Language Arts Writing Standards	The student is able to provide a set of sequential directions for accomplishing a task.	1. Automating solutions through algorithmic thinking. 2. Identifying, analyzing, and implementing possible solutions with the goal of achieving the most efficient and effective combination of steps and resources. 3. Generalizing and transferring this problem-solving process to a wide variety of problems. 4. Tolerance for ambiguity.
2. Growing Plants Interdisciplinary PK-2	Grade 2 Common Core Math Standards	1. Provide students with the tools to overcome negative messaging, challenges, and unknowns. 2. Understand how a plant develops from a seed. 3. The ability to communicate and work with others to achieve a common goal.	1. Formulating problems in a way that enables us to use a computer and other tools to help solve them. 2. Representing data through abstractions such as models and simulations. 3. Automating solutions through algorithmic thinking. 4. Identifying, analyzing, and implementing possible solutions with the goal of achieving the most efficient and effective combination of steps and resources. 5. Persistence in working with difficult problems. 6. Confidence in dealing with complexity. 7. The ability to communicate and work with others to achieve a common goal or solution.

3.	Food Chain Science 4	California, Life Sciences, Grade 4	Students will create an animation representing the food chain using Scratch.	1. Representing data through abstractions such as models and simulations. 2. Automating solutions through algorithmic thinking. 3. Generalizing and transferring this problem-solving strategy to a wide variety of problems. 4. Persistence in working with difficult problems.
4.	Persuade Me Language Arts 5	Common Core Writing Standards, Grade 5, Standard 1 & 4	1. Students will identify the variables in an effective opinion piece. 2. Students will determine criteria for an effective opinion piece. 3. Students will utilize criteria to create a persuasive essay or opinion piece. 4. Students will produce a publishable opinion piece.	1. Logically organizing and analyzing data. 2. Representing data through abstractions such as models and simulations. 3. Identifying, analyzing, and implementing possible solutions with the goal of achieving the most efficient and effective combination of steps and resources. 4. Generalizing and transferring this problem-solving process to a wide variety of problems. 5. Able to handle open-ended problems. 6. Ability to communicate and work with others to achieve a common goal or solution.

Note. Adapted from "Computational thinking teacher resources", 2nd edition, by International Society for Technology in Education (ISTE) and Computer Science Teachers Association (CSTA), 2011, p. 8. Copyright 2011 by CSTA and ISTE.

Computer Science-Specific Resources

Each of the three resources above from Google, ISTE and CSTA focused entirely on CT and provided instructional and PD resources. The remaining three resources target CS, and as such, encompass CT. Thus, elements of CT instruction are still present. Two of the resources, those from Code.org and Brennan, Balch, & Chung (2011) mention CT specifically in their PD-focused teacher resources in order to bolster teacher knowledge of CT. The third resource is unique in its approach to CS instruction and will be addressed in its own section.

Code.org is a non-profit organization committed to promoting equity and expanding access to computer science learning to all groups, but particularly to women and underrepresented minorities. Code.org hopes that every student in every school will have the opportunity to learn computer science, in the same way they can learn about geometry, biology, or chemistry (Code.org, 2019a). Code.org works to promote equity and access through their belief in CS for all by offering free CS curriculum to anyone who wants to learn. Their “CS Fundamentals” course targets grades K-5. Each year, Code.org organizes the “Hour of Code” campaign, an effort which has involvement from approximately 10% of all students across the globe (Code.org, 2019a). Code.org’s instructional resources provide sequenced coding activities and instructional videos that students can work through in order to learn specific elements of coding, such as loops and conditionals. Although Code.org does not focus exclusively on CT, they do address the domain in their resources and include each element of CT identified by Angeli et al (2016).

Many of the programming activities on Code.org make use of block-based coding languages. Block-based coding was popularized by Scratch, a programming environment developed by the MIT Media Lab (Resnick et al., 2009). Scratch is a graphical block-based programming language, where users write programs by snapping bits of code together. Code blocks control different “sprites,” or actors on the screen. Sprites are objects controlled by the written code, like a cartoon cat or a ball bouncing across the screen. By using pre-populated bits of code to snap together, users avoid issues with specific syntax and structure. Code that doesn’t make sense won’t fit together, similar to pieces of a puzzle. Instead of learning specific programming languages and functions, students can focus on CT principles and skills needed to design creative adventures for their sprites.

One identified resource, ScratchEd, focuses on providing instructional resources for teachers who want to use Scratch (Brannan et al., 2019). ScratchEd is a collaboration between researchers at the Harvard Graduate School of Education and the Education Development Center’s Center for Children and Technology. PD provided by the site includes a definitional framework and resources supporting the development of, and assessment of the development of CT. Their definitional framework includes concepts, practices and learner perspectives.

In order to assist with the development of CT, there is a “Creative computing” guide for teachers, consisting of seven instructional units and over thirty activities (Brennan et al., 2011). To aide in the assessment of CT development, ScratchEd provides PD through an interview protocol and rubric for artifact-based interviews, three sets of design scenarios of increasing complexity, and tools for collecting learner documentation.

There are also a number of videos that teachers can watch to see examples of the rubrics in use.

Unplugged Activities

Some resources take a more unique approach to teaching CT and CS principles. These resources, referred to as "unplugged," because they work without the use of any digital technology, often allow for more kinesthetic learning activities where the participants physically manipulate objects to perform their CS or CT-related tasks. These unplugged activities also promote equity and access, as students do not need to have technology in order to practice the skills. Lack of available technology and infrastructure need not prevent students from learning CT and other CS concepts

One resource was created by the CS Education Research Group at the University of Canterbury in New Zealand. The CSUnplugged (Bell et al., 2015) activities book offers free activities that teach CT, available in over twenty languages (Unplugged, 2019), through the use of cards, string, crayon and physical movement, engaging games, and puzzles. The "Beat the Clock activity" on sorting networks builds skills in comparing, ordering, developing algorithms, and cooperative problem solving. In the activity, students are given cards and must sort themselves out into a particular order by following a plotted path and comparing cards along the way (see Figure 6). The basic example shown within Figure 6 shows cards numbered 1-6 in the boxes on the left. Students follow the paths of the arrows so that two students stand in a circle at the same time. Those students then compare their cards, and in this example, the student with the lower number moves to the left (or straight ahead), while the student with the higher number moves to the right (or straight ahead). As students move from one row of circles

to the next, they're continuously comparing their numbers until they ultimately end up in the squares on the right in the correct number order, 1-6. This activity can be modified and repeated to include cards with anything that can be sorted sequentially, such as numbers, words (alphabetically), or Presidents (chronologically). This particular activity from CSUnplugged asks students to follow an algorithm to solve a problem. The CSUnplugged curriculum is available for free on the internet. A full lesson plan is found in Appendix C.

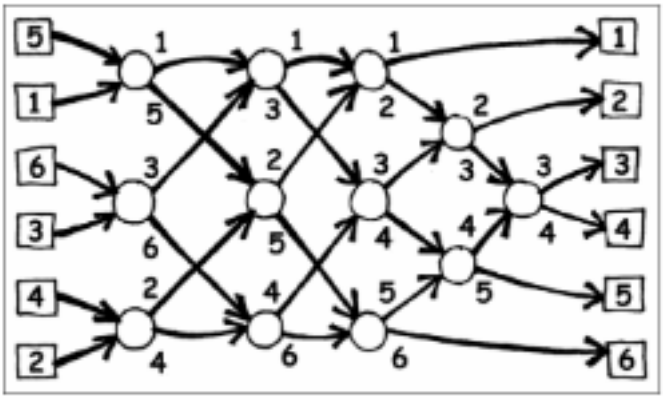


Figure 6: Example of sorting network activity from CSUnplugged lesson (CSUnplugged.org)

Similarly, Code.org's free standardized elementary curriculum in Code Studio has over thirty-five unplugged lessons and activities that reinforce the concepts behind CT (Code.org, 2019b; Prottsman, 2014). In one lesson on real-life algorithms, students work to identify the correct steps to make a paper airplane from a list with some incorrect steps. After identifying the steps, they must put them in the correct order and trade with another person or group to see if their steps, or algorithm, is correct (Code.org, 2019c).

The first lesson in Course 3 of Code Studio is an unplugged lesson on CT. It is designed to be completed as a part of a sequence and is recommended for 4th or 5th

graders. This resource from Code.org is admittedly complicated. The teacher instructional video and the lesson plan say that the students will be discouraged because of the complexity of the task, but one goal of the lesson is for the students to work through that frustration. The lesson plan provides teachers with a tremendous amount of support, and has ways to introduce their CT-specific vocabulary. And while the four terms introduced in this lesson are not completely identical to the elements of CT identified by Angeli et al. (2016).

Code.org does everything that it can to make this lesson a success for teachers. It provides print activities and assessments, easy to follow steps for teachers, and video tutorials for the teachers to learn more about the lesson. There is even another video, "Unplugged Lesson in Action – Computational Thinking" that features the same Code.org instructor from the tutorial video actually teaching the lesson to a class. The Code.org lesson plan itself does not link to this video, but YouTube suggested it to me while I was watching the "Course 3 – Computational Thinking" video. This video could be very helpful to teachers, as they get to see the lesson plan in action and have an expert modeling the instruction for them. According to teachers who have used unplugged resources from CSUnplugged or Code.org, the off-line, hands-on approach to addressing CS concepts is fun and engaging for students who enjoy a more tactile approach to learning (personal communication, December 8, 2017).

Resources for teaching elements of CS that include CT are readily available, but usage data and research on these resources are limited (Buitrago Flórez et al., 2017; Fessakis, Gouli, & Mavroudi, 2013; Israel et al., 2015; Sáez López, González, & Cano, 2016). Code.org tracks and reports the number of accounts created on their site and can

monitor progress made on those accounts. Google does not report the number of educators who engage with either of their platforms for learning about and how to teach CT. Materials from ISTE, CSTA, Computing at Home, K12CS.org and CSUnplugged are not interactive, and those organizations do not report the number of downloads of their resources. Coupled with a lack of data on classroom effectiveness, it is challenging to know if and how these available resources are implemented, making it difficult to evaluate their effectiveness or efficacy.

Resources Strengths and Weaknesses

Each resource above has its own strengths and weaknesses. Code.org advertises a tremendous reach to students and teachers across the globe, with advocacy, equity and access as top priorities. Elementary schools in RCSD participate in the Hour of Code annually, and some teachers direct their students to complete activities on Code.org within their classroom during times of independent choice. But Code.org doesn't teach students specifically about CT, nor does it include instruction in abstraction or decomposition. It does have teacher resource materials that reference CT as one of the benefits of CS education, and elements of CT are present in curricular units provided by Code.org, along with other CS content like internet safety and cyber bullying. It does not provide teachers with specific information about what CT is, why it should be taught, and ways that learning CT can be challenging for students. While ways of teaching CT are provided, teachers are not told that directly. The teacher resources are not lesson plans, but instead sequential activities for the students to complete which will immerse them into coding experiences.

The two resources provided by Google, Exploring Computational Thinking and Computational Thinking for Educators, are valuable resources for expanding teacher content knowledge about CT and have some provided curriculum. The lesson plans seem extremely isolated and might be difficult to pair with traditional content area instruction. “Making Music with Algorithms,” found in Appendix B, is a one-off lesson that integrates algorithms into a music lesson. That is less likely to help a general classroom teacher who doesn’t know how to teach music. The CT course, when offered synchronously for users to complete, would provide a valuable opportunity to interact with and learn from fellow participants and share their created resources. In its current, offline state, participants complete the work in isolation, with no feedback or interaction. The content is still valuable, but the course leaves something to be desired. Further, with no direct links to programming, these resources do not include elements of debugging.

Another resource that excludes debugging, the Computational Thinking Teacher Resource (ISTE & CSTA, 2011b) has detailed lesson plans across all grade levels and includes four out of the five elements of CT. However, there is only one lesson plan for each grade band, leaving the impression that the instruction of CT would be a single, isolated occurrence. This may be the case in some situations, but it is likely not what advocacy groups or the VDOE want for students, any more than teaching science or social studies is done once a year. That being said, the other lessons could serve as examples and be modified by teachers who have developed an understanding of the ways to teach CT to be used in their own grade level.

Creative Computing, the guide for using Scratch in the classroom, is a tremendous resource for using the program in the classroom, but its main focus is on programming

elements of CS instruction. Algorithms in the forms of conditional loops have included instruction, as well as a heavy emphasis on debugging, but otherwise there is little direct instruction on the other three elements of CT. However, not to be overlooked, Scratch is very popular with students and teachers in RCSD.

CSUnplugged works to overcome issues with access to technology and allows students and teachers to develop CS principles without devices. The unplugged activities largely focus on general CS principals like data representation and human interface, but activities on procedures, algorithms, and cryptography teach elements of CT to students in a fun way. The sorting network activity described on page 46 can be repeated with a variety of content areas that involve the ordering of variables (colors on the visible light spectrum, sight words in the dictionary, historical events, etc.). Once students learn the rules to the activity, the critical thinking and evaluation can be applied to any content area, allowing teachers to include the CT activity into other content area instruction as desired. These unplugged activities do not address abstraction or generalization.

The most common element found in the resources in algorithmic thinking, which supports Denning's (2009) assertion that CT and algorithmic thinking are synonymous. It is interesting to note that the three resources most closely associated with general CS principals and programming, Code.org, Creative Computing (Brennan et al., 2011) and CS Unplugged (Bell et al., 2015), are the three resources that include explicit instruction on debugging. Debugging, an essential element of CT, is also essential to the task of programming. The remaining resources have no direct connection to programming, and thus no direct instruction on debugging.

Teachers looking to teach CT to their students have a variety of options available to them online from organizations eager to promote CT and CS education in the classrooms. While no one resource seems to provide teachers with an array of lessons or activities to teach all of the elements of computational thinking, teachers who have gained an understanding of CT and how to teach it might be able to create or modify resources to make CT a regular part of their instructional routine. Teachers can search for resources about CT, or they can look for ways to teach the elements of CT in isolation. If teachers hope to teach CT by addressing all five of its elements, they can design their own curriculum through targeted use of various components of these resources. In order for teachers to evaluate these resources themselves, they must understand CT and the best ways to teach CT. In addition to being able to search the internet for the resources reviewed above, teachers in Virginia can take part in a state-wide PD opportunity to develop CT PCK and learn about instructional resources available for integrating CT into their instruction.

A PD Resource in Virginia

Work is being done within the United States, and specifically in Virginia, to develop teachers' knowledge of ways to teach CT. CodeVirginia, a non-profit group in Virginia working to bring equitable CS education to all of Virginia's students, offers a variety of different training opportunities for free throughout the year. This is in response to the new Virginia SOLs that include computer science, computational thinking, and coding. Two trainings relevant to this capstone study are the Launching Computer Science (K-5) and the Elementary Computer Science Coaches Academy. The Launching CS training is an eight-hour preparation for classroom teachers to integrate CS into their

core curriculum. The Coaches Academy hopes to “train the trainers,” preparing educators to offer PD on CS and CT curriculum. The academy requires more of a time commitment. In addition to an initial five-day training, there are four online modules that require practice in CS work, and an additional requirement of four day-long sessions on Saturdays throughout the school year.

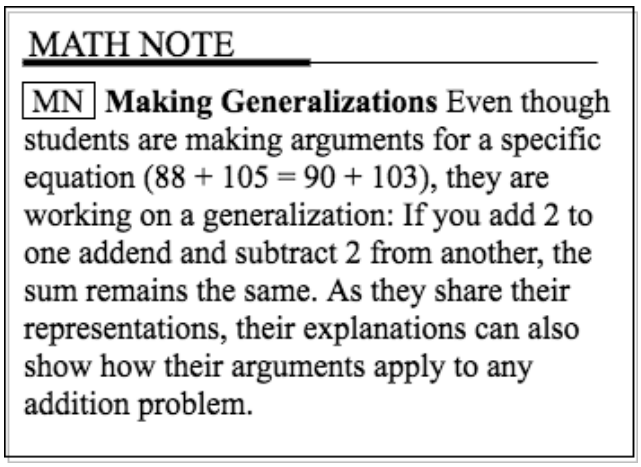
During one week-long session at the Coaches Academy, attendees are introduced to the new VA CS SOLs, spend time unpacking them to identify which standards are already being met in the classroom through other content integration, and which standards will be new to teachers and students. Participants work collaboratively to design lessons that teach teachers CS and also draft lesson plans for teachers to use with students that integrate the new standards into existing content area curriculum. Participants also become acquainted with a number of resources available to teach CS. These recourses include Scratch and Code.org. Through discussion amongst participants, other resources were shared, but they are largely proprietary resources that schools and districts had purchased which incorporated coding into play. Some of these resources are Spheros, Ozobots, Osmos, Parrot Drones, BeeBots, and Makey. The cost of some of these resources put them out of reach for some schools and teachers, and thus their students. Computational thinking was not discussed during this training, although it was mentioned in the required reading by Martinez & Stager (2013). Scratch and Code.org, free resources that can be available to all students, will be reviewed in the following section, along with other free resources. CodeVirginia offered twenty-two such sessions in the summer of 2018 and seven sessions in the summer of 2017, hoping to expand knowledge of ways to teach CS to more elementary school teachers across the state. In

addition to being able to attend this PD opportunity or ones similar, teachers in RCSD are provided with curriculum and pacing materials to guide their content-area instruction.

Investigations Curriculum

One resource provided to all elementary teachers in RCSD is Investigations. RCSD adopted Investigations as a district-wide math curriculum in 2009, and recently transitioned to the third edition of the curriculum in the 2017-18 school year. The goal of the curriculum, published by TERC, is to engage students in conversations about mathematics. Students are encouraged to explain their process, talk to classmates, and share the thinking and understanding behind their actions in solving math problems. The curriculum is more conceptual and less skills driven. The more rigorous third edition of Investigations has 130 lessons across eight units that expand differentiation, provide more extensions, and more remediation than the previous version.

Teachers may use their current curriculum, such as that from Investigations, to incorporate CT instruction in their lessons, and there may be instances of CT instruction present currently. In the third grade Investigations teacher manual, there is a “Math Note” to the teacher about making generalizations (see Figure 7). This note explains to teachers how students are using a CT skill, without specifically mentioning CT. This type of instructional tip might be present elsewhere in the Investigations curriculum or in curricula for other content areas such as science, social studies, or language arts. Curriculum adopted by RCSD and used by teachers to integrate CT was reviewed as part of the data collection in this capstone study and discussed further in chapter four.


MATH NOTE

MN **Making Generalizations** Even though students are making arguments for a specific equation (88 + 105 = 90 + 103), they are working on a generalization: If you add 2 to one addend and subtract 2 from another, the sum remains the same. As they share their representations, their explanations can also show how their arguments apply to any addition problem.

Figure 7: "Math Note", adapted from Third Grade Investigations (TERC, 2016, p. 83)

CT in Elementary School

Very few research studies on computing initiatives focus specifically on CT in elementary schools (Israel et al., 2015). Researchers at Tufts University designed the TangibleK program, a series of lessons involving robotics and programming that targeted CT skills with kindergarteners (Bers, 2010; Bers, Flannery, Kazakoff, & Sullivan, 2014). They found that students who participated in the program scored higher on story sequencing assessments than those who did not participate. In their review of the state of the field of CT, Grover & Pea (2013) make no reference to studies of CT in elementary school, and claim that the lack of funding in computing education causes that void. Lye & Koh (2014) reviewed 27 studies from a search for "computational thinking" in the SSCI and ERIC databases. Of those 27 studies, only 9 were carried out with K-12 students, and only 2 were of elementary students in a classroom setting. Much attention is paid to CS and CT by computer scientists, but research by educational experts is limited (Buitrago Flórez et al., 2017). Research on CT in secondary and higher education has found that teaching programming, and having students writing code, is the best way to teach CT (Buitrago Flórez et al., 2017; Cetin & Dubinsky, 2017). Students are exposed

to CT during programming (Lye & Koh, 2014), but learning to program is hard, and the frustrations that students experience trying to use proper syntax, debugging, and learning a new language interferes with developing CT skills (Buitrago Flórez et al., 2017). One programming language in particular that works to reduce difficulties in programming caused by syntax is Scratch.

CT and Scratch

As described above, Scratch is a visual, block-based programming language designed with novice programmers in mind (Maloney, Peppler, Kafai, Resnick, & Rusk, 2008). The Scratch screen (see Figure 8) and user interface make "the key concepts of Scratch as tangible and manifest as possible" (Maloney et al., 2008, p. 368). Users create programs by dragging blocks from a palette into a script area. The blocks will control designated "sprites," or characters, whose behavior will be seen on the stage. Issues with syntax are addressed by incompatible blocks—the blocks won't fit together if the code doesn't have proper syntax (Sáez López et al., 2016).

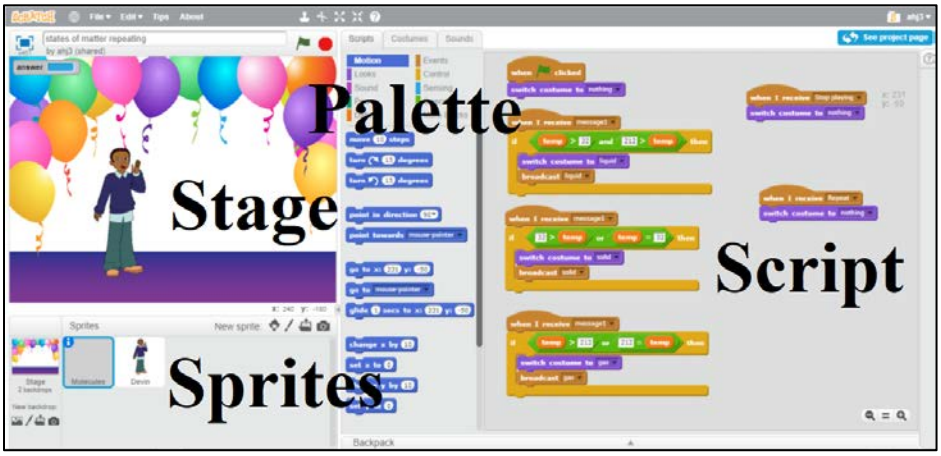


Figure 8: Screenshot of Scratch Interface.

Buitrago Flórez et al. (2017) share a variety of other programming languages used to teach programming to beginners, including LOGO, Python, LEGO® Mindstorms, and

Alice. This literature review will limit the focus to Scratch, however, for two main reasons. First, Scratch is a free, open-source platform and widely supported by a growing body of resources (Israel et al., 2015), including those previously covered such as the K-12 Computer Science Framework (K12cs.org, 2016) and the Computational Thinking Teacher Resources (ISTE & CSTA, 2011). The second reason is that Scratch is installed on every student and teacher computer provided by the Rockview County School District, and it is widely supported by members of the Department of Instructional Technology. The section that follows offers a brief summary of a few research studies where CT was studied through the use of Scratch.

Digital storytelling and gaming. Working with one nine-year-old boy over the course of twenty-four weeks in an after-school program, Lee (2010) sought to investigate whether Scratch could allow the boy to learn CT concepts and skills while creating multimedia products tied to language arts projects. During the first six weeks, the boy learned the fundamentals of Scratch by completing simple activities under Lee's guidance and instruction. However, some of the skills were not mastered during the first six weeks. Over the next eighteen weeks, the boy created digital storybooks or interactive computer games using Scratch but experienced some difficulties due to deficiencies from the first six weeks of the investigation. In particular, the boy had trouble with implementing event-driven programming and understanding variables and their ownership. Despite these difficulties, Lee found that Scratch was an effective programming environment to help the boy learn computer programming concepts and skills while working on technology-enriched language-arts projects. Additionally, the boy found the use of Scratch to be motivating and fun and was excited to create Scratch

projects based on his writing samples. This study is not necessarily generalizable, particularly to implementation on a large scale in an elementary school due to its small size and one-on-one nature in an informal setting, but does show the potential for Scratch as a tool for infusing CT into content area curriculum.

Cross-curricular Scratch and Etoys. On a larger scale, Israel et al. (2015) studied the implementation of a school-wide computing initiative focusing on CT in one Midwestern elementary school. They collected data from seven teachers who volunteered to integrate Scratch or a similar platform, Etoys, into their curriculum; three special area teachers who taught all three hundred students in the school, three classroom teachers across grades 2-5 who taught one class of students each, and one enrichment teacher who worked with a small group of students during pull-out times.

Although the study did not collect student data, they did find through observations and interviews with the teachers that the reported use of Scratch increased the engagement levels, problem-solving abilities, demonstration of initiative and leadership, and overall success of students who participated. Teacher participants also shared that they thought it was important for administration to support their efforts to be innovative as well as how the change would impact other initiatives in the school, classroom management, and teaching practices. Researchers suggested that future studies should look to align the curriculum used by teachers and provided by sites such as Code.org with standards from organizations like the CSTA.

Scratch for CT concepts. In a large case study with five primary schools in Spain, Sáez López et al. (2016) sought to analyze acquisition of basic CT concepts in art and social science through the use of Scratch. Over the course of two years, 107 5th and

6th graders participated in 20 sixty-minute Scratch-based instructional modules focusing on math, science and cultural lessons, as well as specific computational concepts and practices such as sequencing, looping, conditional statements, parallel execution, event handling, user interface design, and keyboard input. Researchers used the Visual Blocks Creative Computing Test (VBCCT), a 40-item test with a "structured and progressive sequence" (Sáez López et al., 2016, p. 134) that assessed the students on CS- related aspects such as the use of different command blocks that demonstrate understanding of the coding process and creation of content-focused computer animations, games and creations. Students participating in the experimental group saw statistically significant improvements of computing concepts and practice scores from pretest to posttest of the VBCCT and outperformed the control group that did not participate in the intervention. Results suggest that curricula using Scratch can help students understand CT concepts.

The above studies highlight the limited research available on integration of CT instruction in elementary school classrooms. Implementations are often small in scale, limited in duration, voluntary and not prioritized by supervising administration. Most studies used Scratch or some other similar visual coding language with students. Future research should focus on how teachers use available resources to prepare to teach CT in their classroom, and what those teachers need to know in order for their students to be successful at CT.

Conclusion

This chapter reviews definitions of CT and what should be taught, why CT should be taught, what teachers need to know in order to teach CT, and current resources

available to teach CT in elementary schools,. The chapter identifies the following key points:

- Importance of CT as a fundamental skill for computer scientists is agreed upon, but support for its value to other domains is lacking (Denning, 2017).
- Discussion of integration of CT into K-12 school curriculum has become a national and international phenomenon among CS educators and advocates (Aho, 2011; Angeli et al., 2016; Bers et al., 2014; Brennan & Resnick, 2012; Catrow, 2016; National Research Council, 2010, 2011; Fluck et al., 2016; Wing, 2010).
- Understanding of best practices for instruction is developed through experience in teaching, collegial exchange of ideas, and reflection on successes and struggles in the classroom (Buchholz, Seli, & Schulte, 2013).
- Collaboration by leaders in K-12, higher education, and industry has resulted in a wealth of free resources for integrating CT instruction and foundations of CS into K-12 curriculum (Armoni & Gal-ezer, 2014; Fluck et al., 2016; K12cs.org, 2016; Kumar, 2014a; Partovi, n.d.; Prottsman, 2014).
- Research on implementation of CT integration in K-12 schools is limited, especially in elementary grade levels (Angeli et al., 2016; Israel et al., 2015; Owen et al., 2014).The Scratch programming language is a popular and somewhat successful tool at teaching CT skills to students.

The literature on CT undoubtedly has gaps, particularly literature related to integration in grades K-6. There is an overreliance on the opinions of thought leaders in the field of CS to justify the value of CT, without empirical evidence to support such claims. There is a small body of research on ways to teach CT. The literature presented

here shows that resources are available for schools to integrate instruction on CT, but there are few empirical studies that investigate those resources or how they relate to standards-based educational practices. Empirical research would greatly impact the collective knowledge on integration of CT, particularly in elementary grades. In order for teachers to develop and improve practices of teaching CT and successfully integrate CT into content area lessons, researchers need to investigate the ways in which teachers make use of available resources, plan for, and teach lessons that specifically target CT skills in elementary classrooms.

Chapter Three

Chapter one provided background of the problem of practice and the conceptual framework guiding this capstone study. Chapter two reviewed the literature relevant to computational thinking (CT) in elementary school classrooms. This chapter discusses the research methodology for this capstone study. Specifically, this chapter addresses the research approach, the research sites and participants, data collection methods, data analysis methods, trustworthiness, ethical considerations, research bias and assumptions, and limitations.

Purpose and Research Questions

The purpose of this capstone study is to understand how elementary classroom teachers integrate CT into instruction. The research questions for this capstone study are derived from the five elements of computational thinking (Angeli et al., 2016) and research on CT. Specifically:

1. How do elementary teachers in Rockview County School District integrate computational thinking into their instructional lessons?
    a) What elements of computational thinking are present in the lessons?
    b) To what extent is computational thinking effectively integrated into the instructional lessons?
    c) What activities or tasks are students engaged in that purportedly teach the elements of computational thinking?

   d) What types of instructional strategies are teachers using to integrate computational thinking into their instructional lessons?

2. How do elementary teachers in Rockview County School District define computational thinking?
3. How do elementary teachers in Rockview County School District prepare to integrate computational thinking into their instructional lessons?
   a) To what extent do teachers use available computational thinking resources when integrating computational thinking into their instructional lessons?
   b) To what extent do teachers use district and school support personnel to plan for instruction that integrates computational thinking into their instructional lessons?

Because this capstone study examined the implementation of new state standards and the ways that teachers implement instruction based on those standards, teachers' planning and enactment of CT lessons was the focus under investigation.

Methodology

This capstone study is structured as an embedded, single-case study (Yin, 2014) of elementary classroom teachers in three schools in one school district attempting to teach lessons on new CT state standards. According to Yin (2014), case study research is most appropriate when "examining contemporary events, but when the relevant behaviors cannot be manipulated" (pg. 12). The research questions seek to answer "how" and "why;" thus, case study methodology is preferable to other methods (Yin, 2014). Yin (2014) argues that case studies should be used when you seek an understanding of an authentic situation, but that understanding will require attention to contextual details.

A case study is defined as an "empirical study that investigates a contemporary phenomenon in depth and within its real-world context, especially when the boundaries between phenomenon and context may not be clearly evident" (Yin, 2014, pg. 16). In this study, the real-world context is the school and classrooms where the phenomenon of instruction will take place. How teachers prepare and instruct and are supported in doing so in one context may be impacted differently in another context, so the research questions are dependent on the contextual setting (Yin, 2014). The case study methodology provides an immersive experience for the reader to provide a better understanding of the relationship between the phenomenon and the context (Marshall & Rossman, 2011; Yin, 2014).

Research Site, Participants, and Access

The focus of this capstone study was three elementary schools in Rockview County School District (RCSD): Rusty Falls Elementary School (RFES), Hilltop Elementary (HES), and Ridge Elementary School (RES). A general overview of both the school district and elementary schools are presented. A comparison of student demographics and total enrollment for RCSD and the three elementary schools is presented in Table 8. Research sites were selected for convenience. Administration at RFES, HES, and RES agreed to be a part of a pilot program in RCSD that implemented new CS SOLs from the VDOE that were published in November, 2017. This capstone study investigated implementation in three of those pilot groups.

Table 8

Comparison of student demographics as percentage of total enrollment as of September 30, 2018

Category	RCSD	RFES	HES	RES
Total Enrollment	14,013	640	482	577
Male	51.1	51.9	53.3	47.1
Female	48.8	48.1	48.1	52.9
Black	10.8	8	24.1	20.3
Hispanic	13.2	13.9	23.2	25.8
White	64.5	64.4	36.7	26
Limited English Proficiency	9.6	10	19.5	32.8
Free and reduced-price	29.4	24.7	53.1	70.9
Students with Disabilities	12.5	10.9	11.6	12.1

Note. "Free and reduced-price" students are those who receive free and reduced-price meals under the federal program. "Students with disabilities" are those identified for special education services, from speech pathology to learning disabilities to severe and profound disabilities.

Rockview County School District

Rockview County School District (RCSD) employs roughly 1,300 teachers with an average of 14 years of teaching experience in a large county in Virginia. Those teachers serve close to 14,000 diverse students, hailing from 89 different countries and speaking 75 different languages. The district's 15 elementary schools (PK-5) spread across 726 square miles of urban, suburban, and rural areas. There is a student-to-computer ratio of 1:1 for students in grades 3-5.

Participant Selection

For the purpose of this capstone study, convenience sampling was used to identify the participants (Marshall & Rossman, 2011). The sampling criteria was that four third

grade teachers volunteer to participate in the study. The participants were selected because they volunteered and because of their convenient accessibility and proximity to the researcher. This type of convenience sampling limits the generalizability of this particular capstone study to the general population, but Yin (2014) argues that case studies are not meant to make statistical generalizations. Case study samples are too small for this common way of generalizing, but instead allow for analytic generalizations, which are opportunities "to shed empirical light about some theoretical concepts or principles, not unlike the motive of a laboratory investigator in conceiving of and then conducting a new experiment" (Yin, 2014, p. 40). Analytic generalizations, principles or lessons learned from case study research may still apply to a variety of other situations in like-cases, or in the case of this capstone study, elementary teachers integrating CT into content-area lessons.

Teachers were asked to participate through consent agreement meetings. I scheduled the meetings through the principals. During the meeting, I told prospective participants about the change in the Code of Virginia which resulted in new Standards of Learning scheduled for implementation during the following school year. I described the expectations of the participants—to teach two lessons on two different content areas that implemented the new standards, particularly those on CT, and also to participate in a follow-up interview after each lesson observation. The consent agreement meetings followed a script (Appendix G) which was approved by the Institutional Review Board (IRB) at the University of Virginia (UVA).

Because RFES had five third grade teachers, I sought participation there first. The principal directed me to reach out to the grade-level leader, who invited me to attend a

team meeting. All five third grade teachers were present, along with the assistant principal. Only two teachers agreed to participate. I next went to HES, which had three third grade teachers. The principal invited me to attend the third grade team meeting, which comprised of all three teachers and the assistant principal. Initially, all three teachers agreed to participate, but two dropped out during scheduling. Finally, I approached RES, where the principal said to contact the grade-level leader directly to schedule the meeting. The grade-level leader invited me to attend a one-on-one meeting to learn about the project, and agreed to participate in the study. As a result of her agreement, I had successfully attained the fourth participant in the study, and did not seek the participation of the other third grade teachers at RES. The sections below describe each research site and the demographics of each participant (Table 9). Investigating how four teachers teach lessons across different subject areas integrating CT allowed for an in-depth look at each individual and the opportunity to compare practices between the teachers.

Rusty Falls Elementary School

Rusty Falls Elementary School (RFES) is located in the northern end of the district. RFES serves a population that is less diverse than some other elementary schools of similar size in RCSD. Compared to the other two research sites, RFES serves over twice as many white students and close to half as many students who receive free and reduced-price meals under federal programs. It has a similar percentage of special education students with Individualized Education Programs (IEP). Student achievement on state standardized tests at RFES for reading and mathematics have improved or been maintained each of the past five years. Each grade level has between five and seven

classroom teachers. The total enrollment of students at the beginning of the school year was around 700. A demographic breakdown of the student body of RFES is available in Table 8. RFES has 110 faculty and staff members serving students in Kindergarten through fifth grade. RFES feed to Central Middle School and River Bluff High School.

Participants at Rusty Falls Elementary School

RFES has a team of five third grade teachers. Three teachers work independently in their own classrooms, while two teachers teach collaboratively. This is not a self-contained classroom with Special Education (SPED) students working with a trained (SPED) teacher to fulfill their Individualized Education Plans (IEP); this is a large classroom with two general education teachers serving twice the number of students as their peers. These two teachers, Sophie Horton and Carmen Sanchez, agreed to participate in this capstone study.

Sophie Horton, a 27-year-old female, came to RFES from the Northeast, where she attended a large public university and earned her teacher licensure and a Bachelor of Science degree in Elementary Education. She is in her sixth year as a teacher, and her fourth year teaching 3rd grade at RFES. Prior to coming to RCSD and RFES, she worked as a K-6 Interventionist and a substitute teacher. Mrs. Horton serves as the third grade team leader at RFES.

Table 9

Participant Demographics

	Susie Jones	Sophie Horton	Carmen Sanchez	Julia Beck
Research Site	HES	RFES	RFES	RES
College	Small Private (1500)	Large Public (98,000)	Small Private (4,000)	Medium Public (20,000)
Degrees Earned	BS Psychology	BS Elementary Ed	BA Art/Spanish, MA C&I ESOL	BA Interdisciplinary Studies, MA Gen-Ed K-3, Early Childhood
Degrees Pursuing	Maintain licensure	None	None	None
Licensure	Through Undergrad minor	Through Undergrad	Through Grad, ESOL, Art, Spanish, General Ed	Through Grad
Current position	Long term sub, 3rd grade	3rd grade	3rd grade	3rd grade
Years of Experience	15	6	7	4
Time in current position	1	4	4	2
Previous assignments	PE, Elem & Mid sub, Interventionist	K-6 interventionist, elementary sub	Photography, Art, 3-4 teacher, sub	1 & 2, long term sub
Additional Responsibilities	None	Grade-level leader	None	Grade-level leader, website coordinator
Age	44	27	27	26
Gender	Female			

Carmen Sanchez, also a 27-year-old female, is from the East coast. She earned a Bachelor of Arts degree in Art and Spanish at a small, private school on the East coast. To acquire teacher licensure, she returned to her *alma mater* to pursue her Masters of Arts degree in Curriculum & Instruction for English as a Second Language (ESOL) with additional endorsements in Art, Spanish, and General Education. This is her seventh year as a teacher, and her fourth year teaching 3rd grade at RFES. Prior to coming to RCSD and RFES, she taught photography, Art, 3rd grade, 4th grade, and worked as a substitute teacher.

Hilltop Elementary School

Hilltop Elementary School (HES) is one of four elementary schools that makes up the "urban ring" of suburban RCSD, bordering a local independent city, Rivertown. HES serves a population that is more diverse than most other elementary schools in RCSD, as is the case with all schools in the "urban ring." Roughly two-thirds of the student population is non-white, and over half of the students are on free or reduced-price lunches. Student achievement on state standardized tests at HES for reading and mathematics has declined each of the past three years, falling from the low 70th percentile to the high 50th percentile. Each grade level has two or three classroom teachers. Additionally, there are two multiage classrooms at HES, one for K-2 students and one for 3-5 students. Each multiage classroom has two teachers. The total enrollment of students at the beginning of the 2018-2019 school year was 482. A demographic breakdown of the student body of HES is available in Table 8. HES has 104 faculty and staff serving students in Kindergarten through fifth grade. All of the students at HES feed to Lafayette Middle School and River Bluff High School.

Participants at Hilltop Elementary School

HES has five third grade teachers. Two teachers work in multi-age classrooms and three teachers each work in their own classroom with one class of students, although the students do switch between those three classrooms for mathematics instruction. Initially, all three self-contained classroom teachers agreed to participate in this capstone project, but two ultimately decided not to participate. The one participant from HES was Susie Jones, a long-term substitute teacher.

Susie Jones, a 44-year-old female, is from the East coast. Ms. Jones earned a Bachelor of Science degree in Psychology from a small private-school on the East coast, and earned her teacher licensure through a minor in Elementary Education. Ms. Jones has fifteen years of experience as an educator, as a PE teacher, substitute teacher in elementary and middle school, and as an interventionist. She works consistently as a long-term sub in RCSD, including two different long-term substitute positions at another RCSD elementary school last year that spanned the course of the entire school year. She has been the long-term substitute of this third grade classroom since the second week of school, when the full-time teacher went out on maternity leave.

Ridge Elementary School

Ridge Elementary School (RES) is also a part of the “urban ring” of RCSD elementary schools. Like HES, RES serves a population that is more diverse than most other elementary schools in RCSD. Roughly three-fourths of the student population is non-white, with a similar number of students on free or reduced-price lunches. Student achievement on state standardized tests at RES for reading and mathematics has dropped over the past six years, falling from the high 50th percentile to the low 40th percentile.

Each grade level has between three and five classroom teachers. The total enrollment of students at the beginning of the 2018-2019 school year was 577. A demographic breakdown of the student body of RES is available in Table 8. Ridge Elementary School (RES) has 123 faculty and staff serving students in Kindergarten through fifth grade. Like HES, all of the students at HES feed to Lafayette Middle School and River Bluff High School.

Participants at Ridge Elementary School

RES has three third grade teachers. All three teachers are self-contained, general education classrooms. One teacher agreed to participate in this capstone project, Julia Beck.

Julia Beck, a 26-year-old female, is from the West coast. She attended a medium-sized public university on the East coast, where she earned a Bachelor of Arts degree in Interdisciplinary Studies. To attain her teacher licensure, she returned to her *alma mater* to pursue her Masters of Arts degree in General Education: Kindergarten through Third Grade and Early Childhood. Ms. Beck has four years of experience as an educator, as a 1st grade teacher in another school district, and as a long-term substitute RCSD. This is Ms. Beck's second year teaching third grade at RES. Ms. Beck also serves as the third grade team leader and the website coordinator at RES.

Data Collection

Data sources for the case studies comprised of audio recorded observations of instruction, post-observation interviews, and documents (Marshall & Rossman, 2011; Yin, 2014). Multiple sources of data were used to increase the credibility of the findings of this capstone study (Marshall & Rossman, 2011; Yin, 2014). Yin (2014) argues that

multiple sources of data collection strengthen the findings or conclusion of any case study.

Observations

According to Marshall and Rossman (2011), observations are fundamental to qualitative research. The observations occurred during instruction of the two lessons that the participant teacher selected to integrate CT.

Lesson implementation. During the classroom observations of teacher implementation of the lessons on CT, the researcher recorded audio and kept field notes. The field notes included data on the setting, the participants, activities, interactions, conversations, and the presence of the researcher (Marshall & Rossman, 2011). The classroom observation protocol, located in Appendix F, was structured to allow for strict observational notes as well as observer comments. The observations were geared towards collecting data for all research questions, but in particular 1a-d and 2a, which focused on instruction that integrates CT. The field notes were be taken by hand and typed the same day. The field notes did not include personally identifiable data to preserve the confidentiality of both the participants and the school.

Interviews

Interviews are the most frequent type of data source collected in qualitative studies (Marshall & Rossman, 2011). The use of a combination of interviews and observations in this capstone study are representative of best practices in qualitative research (Marshall & Rossman, 2011). Participant teachers were interviewed twice, at the conclusion of each observation and the interview questions focused on collecting data for all six of the research questions, but particularly 2a-b. The interviews occurred after the

observation of the lesson taught that integrated CT. The interview was structured but allowed for follow-up questions and clarification. It sought to determine the participants' thinking and understanding of CT and associated value with teaching CT in elementary schools, posed reflection questions on how the teacher planned to integrate CT, and questions on the implementation of the lesson. Questions also gathered demographic information and information about the ways that teachers were supported by school and district-level personnel. Initial interviews lasted between 35 and 75 minutes, and the second interviews lasted between 43 and 108 minutes. The interview protocol for the initial teacher interview is located in Appendix D, and the interview protocol for the second teacher interview is located in Appendix E.

In order to clarify information from early data analysis, I found it necessary to request a third interview with Sophie Horton. Data collection from Sophie Horton's interviews as well as communication from her partner teacher, Carmen Sanchez, led me to request an additional, third interview with Sophie Horton. Ms. Horton agreed to the interview. Those follow-up questions were drafted prior to the interview, and focused on clarifying information from previous interviews and ways that the teachers planned for the instruction of the lessons taught as a part of this capstone project.

All interviews were recorded and transcribed, as is best practice in qualitative research (Marshall & Rossman, 2011). During the transcription, all personally identifiable information was removed, and pseudonyms were assigned to preserve the confidentiality of the participants and the schools.

Documents & Artifacts

A third and final method of data collection was gathering documentation used or produced by study participants during the planning or implementation of the lesson integrating CT, such as the new SOLs (see Appendix F). Collecting documents alongside interviews and observations provides the opportunity for corroboration and augmentation of evidence (Yin, 2014).

For this capstone study, the documents that were collected included lesson plans, emails and text messages to the researcher, instructional resources used in planning, student resources used during the lesson, and any other document pertinent to CT and the CT lessons taught. The documents were logged by date and teacher in conjunction with the observation protocols (see Appendix H). The documents are geared toward collecting data for all research questions, but in particular 1a-b, 1d, and 2a. Any personally identifiable data was removed from the documents reviewed to preserve confidentiality of both the participants and the school. Documents were provided to the researcher by the participating teachers.

Summary of Data Collection Methods

This capstone study used multiple methods of data collection. Data collected was be triangulated to support findings. Classroom observations of lessons integrating CT took place with each teacher twice during the second and third quarter grading periods. Following the observed lessons, the researcher interviewed teachers about the observation at the end of the school day, allowing for teacher reflection. Documents and artifacts were collected during the observation period. Table 10 outlines the relationship between the research questions and how the data was collected.

Table 10

What I need to know, why I need to know it, and how I'm going to get it

What do I need to know	Why do I need to know this	What kind of data will answer this question
What CT knowledge do teachers demonstrate in their teaching and planning?	To identify their knowledge about CT and determine ways to support development of that knowledge	Classroom observations, teacher interview, documents and artifacts
What CT instructional practices do teachers use to teach?	To identify their knowledge about teaching CT and identify ways to support CT instructional practices	Classroom observations, teacher interviews, documents and artifacts
To what extent and how do teachers use instructional resources as they teach CT?	To examine the instructional resources used by teachers as they integrate new content and identify characteristics of successful resources to improve instruction of CT	Classroom observations, teacher interviews, documents and artifacts
To what extent do teachers use district and school support personnel as they teach CT?	To examine the support district and school support personnel used by teachers as they integrate new content and look for ways to improve support to improve teaching of CT	Classroom observations, teacher interview, documents and artifacts

Data Analysis Methods

Data analysis took place simultaneously with data collection and followed the methods outlined by Miles and Huberman (1994). The three concurrent processes offered by Miles and Huberman (1994) are data reduction, data display, and conclusion drawing and verification.

Data Reduction

Data reduction is part of the analysis of data (Miles & Huberman, 1994). Miles and Huberman (1994) define data reduction as "the process of selecting, focusing, simplifying, abstracting, and transforming the data that appear in written-up field notes and transcriptions" (p. 10). This part of analysis occurred continuously throughout this capstone study.

In this capstone study, I began data reduction by coding all observation and interview data with start codes that were developed from the research questions, the theoretical framework and the literature review. Creating start codes from the research questions, theoretical framework and literature review is the method preferred by Miles and Huberman (1994). Table 11 lists the start codes for this capstone study.

I examined the definition of CT that each teacher provided during her second interview. If the teacher offered a definition of CT explicitly during their first interview, I included the definition in the analysis. After asking teachers for their own definition of CT in the second interview, I provided teachers with the elements of CT as described by Angeli et al. (2016) and definitions relevant to elementary school teachers (Table 1, p. 2). After teachers had time to review the table, I asked questions about their instructional experiences with those five elements, specifically. I analyzed responses to explore the beliefs and understandings that teachers have related to computational thinking.

Table 11

Start codes

Description	Code
Computational thinking – Abstraction	CT-Ab
Computational thinking – Generalization	CT-G
Computational thinking – Decomposition	CT-Dc
Computational thinking – Algorithms	CT-Al
Computational thinking – Debugging	CT-Db
Computational thinking – Explicit Instruction	CT-Ex
Computational thinking – Implicit Instruction	CT-Im
Resources – School Support	R-SS
Resources – District Support	R-DS
Resources – Professional Development	R-PD
Resources – Instructional	R-I
Focus – Content	F-Cn
Focus – Computational Thinking	F-CT

Data Display

The reduction and display of data is essential in data analysis as it assists the researcher in processing large amounts of information and better leads to valid qualitative analysis (Miles & Huberman, 1994). Miles and Huberman (1994) suggest the use of matrices, graphs, charts, and networks to “assemble organized information into an immediately accessible, compact form so that the analyst can see what is happening and

either draw justified conclusions or move on to the next step of analysis the display suggests may be useful" (p. 11). The creation and use of displays of data are an essential part of data analysis that occurs throughout the iterative process.

Conclusion Drawing & Verification

Conclusions, often signifying the end, are often drawn before data analysis begins in qualitative research. It is through the data analysis that these conclusions, making sense of the data to derive meaning, are strengthened and solidified (Miles & Huberman, 1994). Those early conclusions were held lightly, as the analysis of the data leads to verification of conclusions. Meanings emerging from the data were tested for their validity through data analysis, and the final conclusions that are drawn were verifiable from the data reduction and display (Miles & Huberman, 1994).

Summary of Data Analysis Methods

Data collection and analysis occurred simultaneously. As data was collected through observations, interviews, and documents and artifacts, it was reduced using start codes. As data was collected and coded, displays were drawn in the forms of matrices, graphs, charts, and networks to assist in the organization of data that led to conclusion drawing and verification. This methodology for data analysis is drawn from the work of Miles and Huberman (1994) and summarized in Figure 9.

In order to ensure that the process of data analysis was clear and logical, I kept an analytic log. This log, adapted from Miles and Huberman (1994), included reflective notes about field notes captured, iterations of patterns and conclusions, as well as the research questions, protocol for engagement with the research participants, and any other comments or notes that increased the quality and trustworthiness of this capstone study.

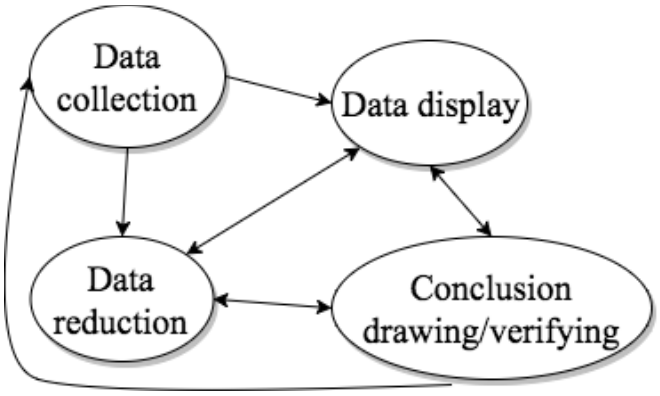


Figure 9: Components of Data Analysis: Interactive Model (Miles & Huberman, 1994, p. 12)

Trustworthiness

Cohen and Crabtree (2006) propose that qualitative research is evaluated on its trustworthiness. Trustworthiness is measured by the constructs of credibility, dependability, confirmability, and transferability (Marshall & Rossman, 2011; Yin, 2014). This capstone study was designed to attain trustworthiness by addressing each construct.

Credibility

Credibility is the "confidence in the 'truth' of the findings" (Cohen & Crabtree, 2006). In order to determine if the findings of a study make sense (Miles & Huberman, 1994), Marshall and Rossman (2011) identify several techniques to maintain credibility in qualitative research: prolonged engagement in the research setting, data triangulation, and peer debriefing. In this capstone study, I spent five weeks at RFES, HES, and RES, observing classroom instruction that integrates CT into content-area instruction, interviewing teachers, and reviewing district curriculum and pacing guides. Data was triangulated through observation notes, interviews, and collection of documents. Lastly, I shared analysis with a disinterested peer, a former doctoral classmate who worked with

me as an ITRT in RCSD, but not at any of the participant research sites. We discussed emerging trends and patterns in the data collection and how those were shaping the conclusions that I was drawing, asking for feedback on the way that I was interpreting the data. This peer debriefing served to ensure that analysis was grounded in the data.

Dependability

Dependability shows that findings of the study are consistent with the data and that the study could be repeated if necessary (Cohen & Crabtree, 2006). Cohen and Crabtree (2006) suggest external inquiry audits to address dependability, while Miles and Huberman (1994) focus on the research questions, study design, researcher's role, data collection, and peer review to strengthen the dependability of the study. In this capstone study, the capstone committee serves as external auditors. Additionally, as previously mentioned, a critical friend helped to provide insight and reliability to the data collection, analysis, and findings.

Confirmability

Confirmability, or external reliability, of a study is the "relative neutrality and reasonable freedom from unknowledgeable researcher bias" (Miles & Huberman, 1994, p. 278). According to Cohen & Crabtree (2006), external audits, data triangulation, and reflexivity support the confirmability of findings in a qualitative study such as this capstone study. External audits were done by the capstone committee and peer debriefing. Data triangulation occurred through collection and analysis of data from multiple sources. Reflexivity was done as I reflected critically on the data and my analysis, to maintain awareness of biases and assumptions throughout the iterative process of data collection and analysis. These practices aided in assuring that the findings

of this capstone study were “shared by the respondents and not researcher bias, motivation, or interest” (Cohen & Crabtree, 2006). The analytic log kept during data collection and analysis was the primary method for maintaining confirmability, in an effort to track any potential bias, motivation, or interests on my part.

Transferability

Transferability, or external validity, is the “ways in which the study’s findings will be useful to others in similar situations, with similar research questions, or questions of practice” (Marshall & Rossman, 2011, p. 252). Providing thick descriptions is a strategy for establishing the transferability of a study (Cohen & Crabtree, 2006). Thick description refers to the level of tremendous detail provided by the researcher in their description of observations collected during the study. In this capstone study, I used thick description when capturing field notes during classroom observations and interviews. These field notes help to ensure the external validity of the capstone study.

Ethical Considerations

Marshall and Rossman (2011) argue that, when evaluating criteria for ethics, “reasoning must move beyond the procedural to focus on matters of relationships—with participants, with stakeholders, with peers, and with the larger community of discourse” (pg. 44). In focusing on the people who participated in capstone study, I emphasized respect for persons (Marshall & Rossman, 2011). Respect for persons requires that participants are treated like people, and not a means to an end. Their privacy, anonymity, and their right to participate was respected. All personally identifiable information was removed from documents and artifacts, and pseudonyms were assigned to each participant. Additionally, this capstone study is subject to the approval of the capstone

committee and the IRB at UVA. As is required by the IRB, each participant was provided with an informed consent form outlining the study and any associated risks with participating in the study (see Appendix I) during the consent agreement meeting. There were no foreseeable risks associated with the study.

For the purposes of this capstone study, I disclosed the purpose of the study to each participant, but not the research questions or any raw data from observation or interview field notes. Information provided to participants during interviews did not include personally identifiable information about other participants. Recommendations made in the final manuscript and Action Communication to RCSD and the VDOE are not specific to teachers, grade-level teams, administrators, or district-level support personnel, but the generalized process and findings.

Finally, all field notes, documents and artifacts collected physically were digitized during this capstone study. At the conclusion of the capstone study, those physical copies will be destroyed. All digital evidence, including audio recordings, transcriptions, field notes, documents and artifacts related to this capstone study will be encrypted and stored in a password-protected file on an external flash drive and locked in a fire-safe box.

Researcher Bias and Assumptions

As a researcher engaging in qualitative research as a participant and observer, I must disclose any biases and assumptions that I have about this capstone study, as they will be impossible to overcome. I am a former elementary school teacher, but not in RCSD. I know many elementary teachers in RCSD both personally and professionally, which will continue after the conclusion of the study. I have a strong interest in CT and see the value in incorporating it into K-12 education. Lessons learned through this

capstone study will better prepare school and district-level support to work with teachers on CT integration.

Limitations

There were several limitations that may have affected the results of this capstone study, some of which could not be fully known until the study was completed. First, participants who volunteered to participate in this study may have done so because of current understandings or classroom practices related to CT. Teachers who did not know what CT is may have elected not to participate. Second, my presence during classroom instruction may have altered the function and actions of the participants. Participants may have felt compelled to put forth atypical effort in order to "look good." Third, findings and recommendations may be transferable to other similar research sites, but the nature of a case study does not lend itself to generalization across all elementary school classrooms in RCSD or to other schools in other districts preparing to teach CT as a part of their curriculum (Yin, 2014). This capstone study only focused on a small number of teachers at three schools in one school district for a limited amount of time.

Summary

This capstone study was an embedded, single-case study of teachers at three elementary schools in one school district who attempted to integrate a new topic into their curriculum. Data collection was done through interviews, observations, and documents and artifacts. Data analysis occurred simultaneously to data collection and was guided by Miles and Huberman (1994). Analysis consisted of an iterative relationship between data reduction, data display, and conclusion drawing and verification. The trustworthiness of the study was addressed through credibility, dependability, confirmability, and

transferability (Cohen & Crabtree, 2006). Ethical matters were addressed through respect for persons and confidentiality. Researcher biases and assumptions were disclosed, and limitations were addressed.

Chapter Four

This capstone study explored how four third grade teachers at three elementary schools in Rockview County School District (RCSD) purportedly integrated computational thinking into their teaching practices. This capstone also investigated how these third grade teachers used resources while planning and implementing lessons that integrate computational thinking instruction. The findings and recommendations that resulted from this case study will provide RCSD with information about its practices to help the organization make informed decisions. The following research questions guided this capstone study:

1. How do elementary teachers in Rockview County School District integrate computational thinking into their instructional lessons?
    a. What elements of computational thinking are present in the lessons?
    b. To what extent is computational thinking effectively integrated into the instructional lessons?
    c. What activities or tasks are students engaged in that purportedly teach the elements of computational thinking?
    d. What types of instructional strategies are teachers using to integrate computational thinking into their instructional lessons?
2. How do elementary teachers in Rockview County School District define computational thinking?

3. How do elementary teachers in Rockview County School District prepare to integrate computational thinking into their instructional lessons?
    a. To what extent do teachers use available computational thinking resources when integrating computational thinking into their instructional lessons?
    b. To what extent do teachers use district and school support personnel to plan for instruction that integrates computational thinking into their instructional lessons?

In this chapter, I first summarize the lessons that I observed at each research site and data collected during the observation relevant to the research questions. Then I provide comparisons between the observations by each teacher. After summarizing each observation, I make comparisons across each participating teacher at all three schools. Following the comparisons, I explore the findings that resulted from my data collection and analysis as they pertain to the research questions. I present implications and recommendations based on the findings in Chapter 5.

At the consent agreement meetings at each elementary school, teacher participants learned of the modification made to the Code of Virginia that added Computer Science and Computational Thinking, including coding, and how that addition resulted in the creation of the new Computer Science SOLs. Each teacher agreed to teach two lessons across two different content areas that integrated CT. After no less than a month from each consent agreement meeting, I was eager to explore how these four teachers had planned for and would integrate computational thinking into their instruction. The eight lessons that I observed were lessons selected by the teacher participants, and teachers planned to address computational thinking specifically in each lesson.

In the next section, I present a description of the classroom observed at Hilltop Elementary School (HES), a description of the lessons that I observed at HES, and a recap of the observations' relevance to the research questions.

Site I – Hilltop Elementary School

I observed one third grade teacher at Hilltop Elementary School, Susie Jones. Ms. Jones taught in a rectangular classroom filled with natural light from a wall made entirely of windows facing southward. Ms. Jones' classroom furniture was comprised of a variety of flexible seating options, including traditional single-student desks, group tables at floor and waist height for students, and seating options away from desks or tables, such as pillows along the window wall. The students in Ms. Jones' class each had a Windows-based laptop computer available to them for use during class, but students did not use these computers during the lessons that I observed. Two of the walls in the classroom had whiteboards, and one of those two walls had an interactive whiteboard as well. The ceiling had a stem-mounted projector for displaying video to the interactive whiteboard. Both observations with Ms. Jones occurred in the morning. During both lessons, in addition to Ms. Jones, another teacher was present to provide individual services for one Special Education student. That teacher did not engage with other the teacher or other students in the classroom during either observation.

Observation I

The first observation with Ms. Jones was in the fall of 2018 and lasted for 25 minutes. There were 17 students present, and Ms. Jones taught the lesson to the whole group. Ms. Jones planned a lesson on computational thinking, specifically focusing on what she called "decompose." In her lesson, she introduced and defined CT as "a way to

understand and solve problems by breaking them down and creating steps to solve them" (Jones, first observation) and "decompose" as "to break things down" (Jones, first observation). She engaged the students in two different examples about the new vocabulary in an attempt to make real-world connections: making a pizza and getting ready to go to school. She asked her students, "How do I make the pizza… what's my first step" (Jones, first observation) and students gave various responses, from using flour to make the dough all the way to putting it into the oven. Students created steps to solve the problem of making a pizza, although Ms. Jones never identified this element of CT, algorithms, by name.

After the vocabulary lesson, Ms. Jones projected a web-based BrainPOP Jr. (2019) video that the students watched for six minutes. The main characters, Annie and Moby, reiterated Ms. Jones's definition of CT. This video specifically named and defined debugging, although Ms. Jones did not reference debugging in her lesson. After the video, students completed an activity where they attempted to explain a multi-step handshake to a partner using small steps. Her directions to the students were brief:

> So you're going to take a problem right now, which I'm going put you with pairs, you are gonna make a handshake that's four steps, you're gonna teach your partner a four step handshake… And then we're going to talk about how you taught 'em (Jones, first observation).

Mrs. Jones told her students that they would have five minutes to complete this activity, but she only gave each partner ninety seconds to teach their four-step handshake. This did not allow for much of an opportunity for the students to engage in breaking down the handshake and teaching their partner the steps to perform it. At the conclusion of the

activity, Ms. Jones reviewed the vocabulary with her students by reminding them that "decompose means to break things down. So today, we broke a few things down to learn about computational thinking. We learned something new" (Jones, first observation). Then the lesson ended.

In this first lesson that Ms. Jones invited me to observe, she intended to teach CT vocabulary to her students explicitly and have them complete activities where they learned the skills associated with that vocabulary. During her post-observation interview, she described her goals of the lesson:

> What I was hoping to teach was decompose. That was the main goal. That they would learn to break down a task into smaller steps. And to create steps, to know that there were steps to solve part of the problem (Jones, first interview.)

She only intended to teach and engage her students in decomposition. She did not indicate that she taught algorithms by name, nor did she recognize that she introduced debugging to her students through the BrainPop video. She only wanted her students "using computational thinking and decompose" (Jones, first interview). She did not plan for a lesson that integrated CT into academic content.

Observation II

The second observation with Ms. Jones was on a Friday two weeks after the first observation. The observation lasted for 25 minutes. There were 15 students present, and Ms. Jones taught the lesson to the whole group. Ms. Jones taught a lesson on cybersecurity, specifically focusing on creating strong passwords. In her lesson, she introduced the need for strong passwords on computers, engaged the students in

situations where strong passwords might be necessary and when having a “bad” password would cause problems:

> We do not want to use easy words about you that people would be easy to guess. For example, if, if, if Doug is a swimmer, ok, and everybody knows that all Doug does is swim. He swims five days a week and summers, on the weekends, six days, maybe seven days, he's always swimming, swimming, swimming, swimming. Do we want his password to be "Doug Swimming (Jones, second observation)?

Ms. Jones also reviewed a list of guidelines for creating strong passwords (See Appendix J) and passed out a worksheet called “Smart Passwords?” (See Appendix K) for students to complete. She circulated the room and guided the whole group through completing the worksheet, where they created passwords for two children based off of their personal interests and the password guidelines: “Right, so, right, it'd be hard to figure out, so that's a good one. So the answer is yes. And you can write how she, we just answered those questions. How did Krystal choose her password” (Jones, second observation)? After students completed the sheet, they reviewed their answers as a whole group and Ms. Jones collected the worksheets.

In the second lesson that Ms. Jones invited me to observe, in an effort to teach CT, she taught a lesson on password safety. Ms. Jones said that “to do computational thinking with, deals with computers, which has to do with making sure they have safe passwords, so they can do the computational thinking” (second interview). According to Ms. Jones, her students were engaged in learning CT in the way that they were thinking:

> They were thinking, it was mostly about passwords. So passwords is what they were thinking about. How they could use it and how they wouldn't want people to break their password because [of] what they could get into and what they couldn't get into on the computer. How they could keep things safe. It was more like the safety of it (Jones, second interview).

She found the lesson to be a success, noticing that her students were "kind of excited about the passwords and they actually understood, because it's something that they use" (Jones, second interview).

Summary of observations of Susie Jones. Susie Jones at HES taught two lessons where she intended for her students to engage in CT. In her first lesson, she introduced CT vocabulary and attempted to engage her students in discussions and activities where they decomposed and created algorithms. Students had very little time to learn or engage in CT experiences. In her second lesson, she attempted to teach her students the value of having strong passwords and methods to keep their passwords safe. There were no elements of CT referenced in her second lesson. In the next section, I present a description of the lessons observed at Rusty Falls Elementary School (RFES).

Site II – Rusty Falls Elementary School

I observed two third grade teachers at Rusty Falls Elementary School, Sophie Horton and Carmen Sanchez. Ms. Horton and Ms. Sanchez team-teach together in a large classroom that takes on an irregular shape because it was a repurposed office space after school renovations. Figure 10 shows a diagram of the room. All four of the observations at RFES took place in the northern section of the room. Ceiling lights in the hallway illuminate the northern section of the room through the windows that make up the top-

half of the walls around the room. The lights in the classroom were off during all four observations. The focal point of this section of the room is the 65” interactive display on a mobile cart. Students who participated in the lessons that I observed sat on the floor in front of this display, while the teacher sat in a chair to the left of the display, facing the students. All students in the classroom had a Windows-based laptop computer available to them for use during class, and the students observed during the four observations all used their computers for the entirety of the lessons. During each observation, the other teacher was in the room but working with other students out of view of the researcher and the lesson participants.

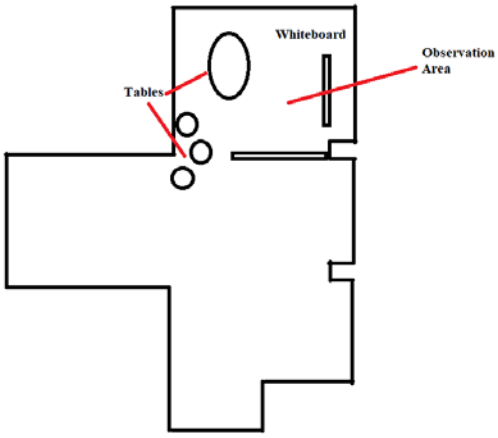


Figure 10: Diagram of classroom in Rusty Falls Elementary School

Observation I – Sophie Horton & Carmen Sanchez

I observed Ms. Horton and Ms. Sanchez at RFES two times each, individually. However, the content taught and instructional strategies used by each teacher during their first lessons were the same. Ms. Horton’s observation was on a Friday and lasted 25 minutes. Ms. Sanchez’s observation was at the same time on the following Monday, and it lasted 30 minutes. Both teachers taught lessons to a small group of students (four by Ms. Horton, six by Ms. Sanchez) on synthesizing text about Ancient Egypt. Teachers

guided students in reviewing notes that they had previously written about Ancient Egyptian Pharaohs and transferring the main idea of those notes onto a slide show on Google Slides, a cloud-based presentation platform:

> And what we're going to do today, since we have synthesized one paragraph from our article, we're going to write that one sentence that we wrote in our research folders. We're going to type that into our Google Drive document (Horton, first observation).

Students had to add a picture to the slide by searching within Google Slides and add captions with different styles of fonts:

> So I really like how Erik has a picture here. Somebody had, oh, and I see that umm, Erik added a little, a brief caption here too. I would like to challenge everybody to add at least one picture to your slides, to each of your slides, and to add a caption to explain the picture (Sanchez, first observation).

Teachers observed students working independently and provided support as needed, either by getting up from the seat and moving to the student to see what they were doing or needed help on (Horton) or by displaying the student's presentation on the interactive display and speaking to them from their seated position (Sanchez). The observations concluded with the teachers telling students that they were out of time, and that they would continue working on this later.

In the post-observation interview with Ms. Horton, she said that her lesson addressed CT through technology integration:

> Well, I would say, from my understanding of what computational thinking is, that we integrated technology into our lesson on this topic of Ancient Egypt, in a way

> that we weren't trying to do too many skills at once… We had already synthesized our information and we were just simply focusing on the digital side of putting that information into another format. So, I would say that they had to, they had to navigate the digital tools of Google Drive through that lesson in various ways (Horton, first interview).

Ms. Sanchez shared a similar notion as her partner-teacher, reporting that students engaged in CT by "writing on the computer, creating a new project from something else" (Sanchez, first interview). Both teachers taught the same lesson and shared similar ideas about the ways that they integrated CT into the lesson.

Observation II – Sophie Horton

My second observation with Ms. Horton at RFES was on the first Wednesday after her previous observation at 9:50 AM. The lesson lasted for 27 minutes. Ms. Horton led a small group of six students through a lesson on graphing using Google Sheets, a cloud-based spreadsheet platform. She began by showing a bar graph to the students and asked them questions about what information could be determined from the graph:

> And it's cool because if I take my mouse and I actually hover over Hot Dogs up top here, it should... yeah, there we go. How many people ate hot dogs, at, let's say it's at a fair. How many people ate hot dogs at the carnival or the fair (Horton, second observation)?

Students completed multi-digit addition and subtraction problems with teacher guidance aloud on the interactive display using formulaic problem-solving strategies, such as "part-part-total." As a group, they created their own bar graph using made up numbers and categories. She said to her students, "Ok. So let's come up with three new categories that

we want to make together. Can someone think of a topic that you feel like might be something is interesting to third graders right now" (Horton, second observation)? Ms. Horton walked the students through the steps of using Google Sheets to create a table with the categories and numbers and convert that table into a bar graph:

> Alright, so now, for us to make the graph, what we have to do is on your mousepad, and this probably would be easy to do with two hands, you're going to click in A1 and drag your fingers so that you highlight in the little, it kind of turns like a blue grey, you're going to highlight through B3, so that all of your numbers and words are highlighted and it will look like this (Horton, second observation).

She concluded the lesson by telling students that it was time to stop and that they would return to this next time.

When I asked Ms. Horton what about the second lesson specifically addressed CT, she said that she wanted her students to learn "all the ins and outs of the spreadsheet, and how you can, you can do more with it than what you might think because there's a lot that's not seen that you can do" (Horton, second interview). She also said that the students were engaged in CT because they were "using the screen, so that was another piece of technology" (Horton, second interview), referring to the interactive display that she used to show her computer screen to the students. She continued, "So they were able to look at the screen during the lesson in order to see the sample graph that I created, and in order to analyze the graph" (Horton, second interview). Ms. Horton did not refer to any elements of CT when sharing her goals of CT integration in this particular lesson.

Summary of observations of Sophie Horton. In the two lessons that Ms. Horton taught, there were no references to elements of CT present in either lesson. She attempted

to engage students in CT through technology-enhanced lessons using Google Slides and Google Sheets. Data from Ms. Horton's interviews did not contain references to the elements of CT.

Observation II – Carmen Sanchez

My second observation with Ms. Sanchez at RFES was on a Thursday at 1:15 PM, ten days after her first observation. The observation lasted for 45 minutes. Ms. Sanchez walked a small group of six students through a paper-based resource from the book *Coding Games in Scratch* (Woodcock, 2016). Each student had their laptop and a nine-page color copy of the section in the book on "How to build Star Hunter," a 41-step guide to create a game using Scratch (See Appendix K). At the beginning of the observation, Ms. Sanchez led the students to open Scratch on their laptops and provided a brief explanation of the layout of the program:

> These are our pieces of code that we're going to be using here. So if you look here at the blue blocks, these are the blocks that will help your sprite move. So that's why it's titled motion here. Click on where it says looks. These are the purple blocks. And this will change how your sprite is, what it looks like. So you can change its color, its size, things like that (Sanchez, second observation).

Then Ms. Sanchez and the students worked through the steps together. Ms. Sanchez's instruction came almost entirely from the photocopied guide as she read the steps synchronously with the students, "We're going to go to step 8. That's the purple step on page 24. And this is where we get to rename our sprite so that it doesn't just stay Sprite2" (Sanchez, second observation).

During the observation, the ITRT at RFES, Catherine Day, entered the classroom and joined the students on the floor, offering support to them as they worked through the steps. Ms. Sanchez also spent some time circulating the room, checking in on the students as they worked:

> Did you, maybe you didn't attach your sound? [Troubleshooting for student silently] Hit ok. That's because, ok, Lula, is yours linked? Yeah, you might need to go to sounds. It says bubbles, oh, yours is playing. Yeah, you can hear it. Is it? Oh, you might need to do the volume up here (Sanchez, second observation).

She modeled following the step-by-step directions on the handout but the students were primarily engaged in following the steps of the handout themselves and did not pay much attention to their teacher. If they encountered an issue, they might ask either of the adults for help, but Ms. Sanchez did most of the talking throughout the observation, reading the handout aloud for the students, "Look on step 12. You should be looking on your paper. What three blocks do you see that are making up step 12? What color is that" (Sanchez, second observation)? After the students got through Step 17, Ms. Sanchez informed them that this was all that they had time for today, and taught them how save the program file to their desktop. After she visually verified that everyone had saved their files, students transitioned to the next classroom activity.

During the second interview, Ms. Sanchez stated that she aimed for her lesson on Scratch as a way to integrate CT into her instruction. She intended for her students to engage in thinking "like a computer, following steps like a computer would and coding in general. And having kids work through a sequence of events to make a certain action happen on their computer" (Sanchez, second interview). According to Ms. Sanchez, this

second lesson addressed CT because students had to "think like a computer, like, so they had to make sure all their code was lined up correctly" (Sanchez, second interview). She attempted to engage her students by leading them through the directions on "How to build a Star Hunter" game.

Summary of observations for Carmen Sanchez. The first lesson that I observed for Ms. Sanchez where she attempted to teach CT engaged her students in creating a Google Slides presentation based off notes on Ancient Egypt. There were no elements of CT referenced in this lesson, nor did Ms. Sanchez reference the elements during her first interview.

The second observation for Ms. Sanchez was a teacher- and handout-led creation of a game using the Scratch coding program. There were no elements of CT explicitly named or referenced by Ms. Sanchez in this lesson. Students had the potential to engage in the use of algorithms and debugging while using Scratch, but did not. In the next section, I present a description of the lessons observed at Ridge Elementary School.

Site III – Ridge Elementary School

I observed one third grade teacher at Ridge Elementary School, Julia Beck. Ms. Beck taught in a trapezoidal room with two large windows in the back and a variety of seating options for the students, including group tables of a variety of shapes and heights. Students could stand or sit in chairs, stools, on pillows or on the floor at the various different tables. The students in Ms. Beck's class each had a Windows-based laptop computer available to them for use during class, but the students did not use these computers during the lessons that I observed. In the corner of the room was an interactive whiteboard and a projector, with a carpet under the projector for students to sit on while

engaged in content displayed on the board. There was some blank whiteboard space next to the interactive whiteboard, shelves with instructional materials covered the remaining wall space. Both observations with Ms. Beck occurred at 1:30 in the afternoon. During both observations, in addition to Ms. Beck, another teacher was present to provide individual services for one Special Education student. That teacher did not engage with other the teacher or other students in the classroom during either observation.

Observation I

The first observation with Ms. Beck was on a Tuesday. The observation lasted for 50 minutes. She taught the lesson to the whole group of 16 students. Ms. Beck began the lesson by projecting a graphic to the students on computational thinking (see Figure 11) that defined the terms “decomposition,” “abstraction,” “pattern recognition,” and “algorithmic design.” Ms. Beck then hung up a hand-made poster with those same terms, a pronunciation, a simplified definition, and an example (See Figure 12). She spent the first ten minutes of instruction reviewing these new vocabulary words:

> Where have you got or had a problem before and you've need to breaking it down into smaller pieces? I’ll give you 30 seconds of think time. When have you had a big problem and you've had to break it down into smaller chunks to be able to solve it? I’ll know you're ready when you have a thumb on your heart (Beck, first observation).

Ms. Beck used instructional strategies such as “mirror me,” where students repeated after the teacher and “turn-and-talk,” which allowed students to talk to their classmates about their understanding of the new vocabulary.

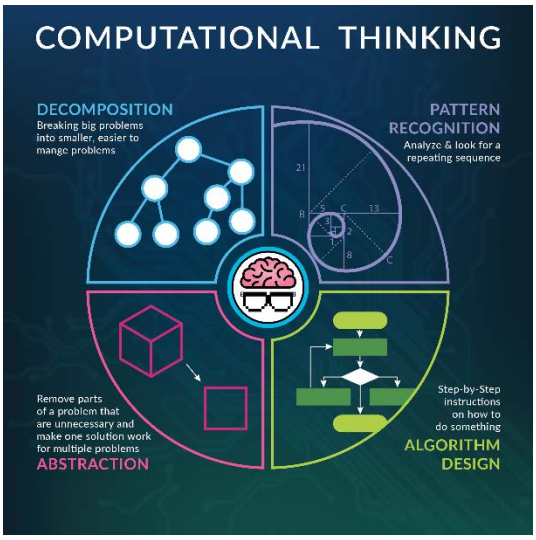


Figure 11: Computational Thinking instructional graphic from ComputationalThinkers.com

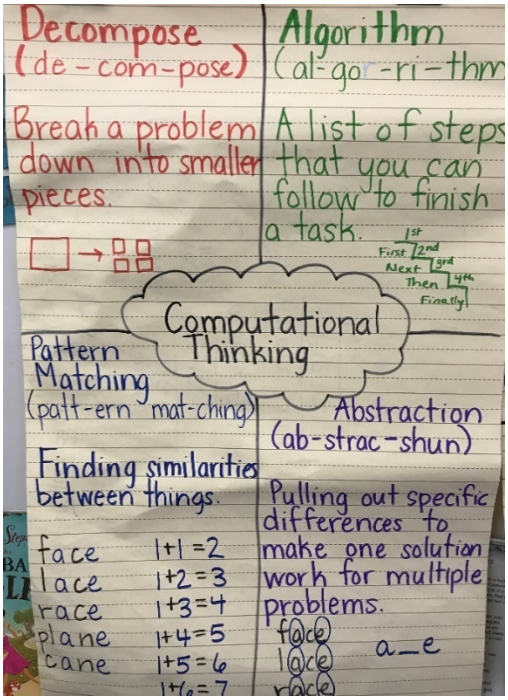


Figure 12: Computational Thinking chart paper definition with examples and pictures

Over the next ten minutes, Ms. Beck explained to students that they would be playing “The Game with No Rules,” distributed associated materials, and helped students to get organized in order to begin the activity. The “Game with No Rules” involved

students using the Computational Thinking Kit from a Code.org unplugged activity on computational thinking (See Appendix M). Following instructions on the kit, students were supposed to figure out how to play a game by examining examples of previous play by fictional students. Ms. Beck explained the game with an example:

> Each player said "I chose a" and then they chose something. Ok. This one, the Player One chose a lion, Player Two chose a donkey, and Player Three chose a puppy. Huh, interesting. All different things. So what am I going to do with lion, donkey and puppy? If I'm circling what's the same, what am I going to do with what's different? What did I say? The instructions are on the board. What am I going to have to do with what's different? You're circling what's the same, what are you going to do with what's different (Beck, first observation)?

Ms. Beck uses the terms "pattern matching" and "abstraction" as they are included in the instructions on the kit. For the next twenty minutes, Ms. Beck circulated the room, visiting the small groups of students who were working to play "The Game with No Rules." Students constantly requested support from the teacher, asking questions about what they were supposed to do and if they had completed the sheet correctly, eager to move onto the next steps where they got to draw pictures. For example:

> Ms. Beck: Alright, so look up here. You're not writing anything yet, you're only circling and underlining. So write your name at the top.
>
> Student: I'm choosing a cheetah.
>
> Ms. Beck: So you're not choosing anything yet, because you are circling and you are underlining. So these, these are already done for you. These are the same. So if they're different, you underline. And if they become the same, then you circle

> again. You're looking for patterns. What don't you understand? So those are all the same, and you underline them. If they're the same, you circle them. Did you hear my words (Beck, first observation)?

Most groups asked many questions about what they were doing and appeared confused about what they were supposed to be doing with the kit from the onset of the activity.

To wrap up the lesson, with ten minutes left, she asked the students who have finished to turn in the kit to the language arts bin, or to put it into a folder to complete it later independently. The students returned to the carpet where they reviewed the four new CT vocabulary words they learned and she repeated the definitions to them: abstraction, pattern matching, decomposition, and algorithms. She asked the students what they did during the activity to engage in each of these four components of computational thinking. When asked how students "did a list of steps" (Beck, first observation), one student offered this explanation:

> Like you would, at first you rolled the dice, and then you next you do, write what you did and then you, then you could, I thought you could actually draw the thing on the animal. Finally you could show the teacher (First observation).

At the conclusion of the lesson, she asked students to talk to a partner about what they learned today. She circulated the room, listened to their answers and asked questions about their responses. One student said that he learned "you could separate it by circling it and underlining" (First observation). When speaking to Jodi and Geneva, she reminded them of the task at hand:

> Ms. Beck: We're talking about one thing you learned when you were playing this game with no rules.

Jodi: I learned... that you can draw things on animals.

Ms. Beck: Sure, yeah. What'd you learn Geneva?

Geneva: I was going to say the same thing as Jodi (First observation).

Finally, she called on two students to share what they had learned: neither student said anything related to computational thinking.

During her post-observation interview after this first lesson, Ms. Beck shared that intended for students to engage in the CT that she defined in class: Abstraction, decomposition, pattern matching, and algorithms. Ms. Beck attempted to teach CT vocabulary explicitly in this lesson. She tried to engage students in learning CT through vocabulary instruction and by completing activities to practice abstraction and generalization through the Computational Thinking Kit, but student did not understand what to do.

Observation II

The second observation with Ms. Beck was on Tuesday one week after the first observation. The observation lasted for 45 minutes. Ms. Beck taught the lesson to the whole group of 16 students. Ms. Beck began the lesson with her students seated on the carpet to review their computational thinking vocabulary words that they learned last week (see first observation above):

Ms. Beck: So in the morning message this morning, I said we were going to review some vocabulary. Raise your hand if you remember what kind of vocabulary we're going to review. Luis?

Luis: We were going to learn about algorithms.

Ms. Beck: We were going to learn about it for the first time or are we gonna review?

Luis: We're going to review algorithms.

Ms. Beck: Ahh, nice. So Algorithm was one of them. What else (Second observation)?

She redefined the terms and asked students to talk to partners about how this vocabulary applied to the real world. For the first ten minutes, she circulated around small groups on the floor to engage with students as they discussed the CT vocabulary. Students shared that they might use these CT skills when playing video games or making fish food.

After ten minutes, she grouped students up and explained to the class that they would be using their knowledge of folk tales and the activity they did last week, the "Game with No Rules," to create a story game, the "Create-Your-Own-Story ROUGH DRAFT" (see Appendix O). She spent five minutes reviewing different types of folk tales before distributing a packet to each student that they would use to create the story game. During the next thirty minutes, groups of students worked to create lists of six characters, settings, problems, and solutions that would be found in their groups' selected type of folktale while Ms. Beck circulated the classroom, supporting students by explaining directions and offering encouragement on the choices that they had made based on their selected folktale. None of the student groups completed their lists during the observation, so they never got to the computational thinking part of the activity. The lesson ended rather abruptly when Ms. Beck rang a bell to get students' attention without warning and told students that it was time to turn in their packets to her and clean up the room.

Ms. Beck described the lesson as “a mix of computational thinking as well as folk tales” (Beck, second interview). In addition to reviewing the language arts content that students have learned related to folktales, she wanted to make CT relevant to her students:

> [This lesson gave] students an opportunity to practice all of their computational thinking skills that we’ve been using. So basically, the computational vocabulary we’ve been working on, I wanted them to be able to apply that to real world situations. That’s why I started the lesson off with talking about how they use it in the real world (Beck, second interview).

Besides reviewing the same CT vocabulary from her first lesson to make real-world connections, students spent all of the activity time building their lists of story elements for folk tales. Ms. Beck recognized that her lesson fell short of her objective. She lamented the lesson “took longer than anticipated, so they didn’t get through the whole lesson. They got through maybe a third of it today… Originally, I had hoped to have a closing time where we can reflect on what we had done” (Beck, second interview).

When asked how the lessons that she taught for this capstone project integrated CT, she listed her four component parts and described the ways that students were doing those things by reiterating what the definition was. For example, the ways students were using algorithms in the first lesson:

> Being able to reword it in steps, like when Steve reworded in steps and said ‘Well, first you pick and animal, and then you roll the dice, and then,’ so he could already tell the progression of how it need to be. (Beck, first interview).

She gave another example for decomposing:

> I think one of my students, Luis, said it really well. He said 'Well just getting this problem was really complicated, but then when you said 'Well, just circle what's the same and underline what's different,' that was a small problem that they could… it was more manageable to do than just like, 'Here, figure this out. Giving them an actual game with no rules, then that would be really tricky because then you like, have to figure out all the pieces, but giving that like, brief step into like, okay, this is what you do first (Beck, first interview).

Ms. Beck was aware of the ways that some of her students were engaged in computational thinking the way that she defined it and the way that she had planned for it. Ms. Beck attempted to teach abstraction, generalization, decomposition, and algorithms through two lessons, albeit with slightly different terms.

Ms. Beck taught two lessons; one in which she tried to teach computational thinking vocabulary explicitly and provided students with an activity intended to engage them in a computational thinking task; and one that focused on language arts content but included a review of CT vocabulary from the first lesson. She intended for students to engage in CT skills through an activity that was similar to that of the first lesson, but students did not have enough time to reach that point in the lesson's activity.

Summary of Findings

The problem of practice that I sought to address through this capstone study was how to help teachers implement new standards for CT and infuse the instruction into their lesson. This capstone study explored how four third grade teachers in RCSD would plan for and teach two lessons that integrated CT into their instruction. The findings of the case study are the outcome of careful, rigorous analysis guided by the research questions

(Miles & Huberman, 1994). In this section, I present the findings (see Table 12) and the data analysis that led to them.

Table 12

Summary of Findings

	Finding	Research Question
1	Only three of the eight lessons taught by teachers in RCSD as a part of this capstone study contained elements of CT. Each of the three touched upon one or more element of CT identified by Angeli et al. (2016) to varying degrees and collectively touched upon all five elements.	1a-b
2	Teachers whose lessons contained elements of CT used direct, didactic instruction in order to integrate CT into their instruction. Teachers focused on helping students understand CT terms and vocabulary by relating the concepts to students' everyday lives.	1c-d
3	Teachers in RCSD did not have a common, shared understanding of the meaning of CT as defined by the elements identified by Angeli et al. (2016) or otherwise. Teachers suffered from definitional confusion related to CT and struggled to make sense of their own interpretations of CT, even when provided with concrete definitions and relevant examples.	3
4	The lessons that touched upon elements of CT were taught by two teachers who used CT resources. Resources include lesson plans, articles, graphics, and instructional videos. Two teachers who did not touch upon elements of CT did not use CT resources.	2a
5	One teacher who taught lessons that touched upon elements of CT used district support personnel. Those personnel accessed and modified CT resources and co-planned the lessons with the teacher.	2b

Finding One: Only three of the eight lessons taught by teachers in RCSD as a part of this capstone study contained elements of CT. Each of the three touched upon one or more element of CT identified by Angeli et al. (2016) to varying degrees and collectively touched upon all five elements.

The first finding of this capstone study relates to the elements of computational thinking (Angeli et al., 2016) that were present in the lessons observed, which addresses

research question 1a-b. These questions inquire about the elements of CT present and the extent to which they are integrated into the lessons. Only three of the eight lessons taught by teachers who participated in this capstone study directly addressed elements of CT. Within those three lessons, I identified at least one instance of instruction for each element of CT. Table 13 provides a summary of the number of elements coded per teacher in each lesson and interview.

Susie Jones taught one lesson where she attempted to teach decomposition vocabulary explicitly. Her lesson also contained implicit instruction of algorithms and an instructional video that she showed her students had explicit instruction in debugging. Julia Beck explicitly taught abstraction, generalization, decomposition, and algorithms vocabulary to her students in each of her lessons. Sophie Horton and Carmen Sanchez did not integrate CT into their lessons.

Table 13

Instances of start codes related to CT instruction referenced in lessons and interviews

Start Code	Jones				Horton				Sanchez				Beck			
	L1	I1	L2	I2	L1	I1	L2	I2	L1	I1	L2	I2	L1	I1	L2	I2
CT-Ab	0	0	0	0	0	0	0	0	0	0	0	0	6*+	2	3*+	2
CT-G	0	0	0	0	0	0	0	0	0	0	0	0	6*+	3	4*+	2
CT-Dc	17*+	4+	0	1	0	0	0	0	0	0	0	0	5+	1	3+	2
CT-Al	10*#	1#	0	0	0	0	0	0	0	0	0	1	5+	1	4+	2
CT-Db	2^	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0
CT-Ex	7	3	0	0	0	0	0	0	0	0	0	0	2	1	4	1
CT-Im	10	1	0	0	0	0	0	0	0	0	0	3	1	0	3	0
F-Cn	0	0	1	0	1	1	1	1	1	1	1	1	0	0	1	1
F-CT	3	3	0	0	0	0	0	0	0	0	0	0	4	2	1	1

Note. Start Code definitions found in Table 11. L1 = First Lesson; L2 = Second Lesson; I1 = First Interview; I2 = Second Interview; * = Student Engagement; + = Teacher Provided Definitions; ^ = Video Provided Definitions; # = Referenced Indirectly.

Finding Two: Teachers whose lessons contained elements of CT used direct, didactic instruction in order to integrate CT into their instruction. Teachers focused on helping students understand CT terms and vocabulary by relating the concepts to students' everyday lives.

The second finding of this capstone study explores the instructional methods, activities and tasks that teachers used to integrate computational thinking into their instruction, which addresses research questions 1c-d. Of the two teachers who included elements of computational thinking into their instructional lessons, the primary method of instruction was through direct, didactic instruction that focused on helping students understand CT terms and vocabulary, as well as relating the concepts to students' everyday lives. For example, Ms. Jones told her students the vocabulary that she wanted them to learn and defined it for them. She directed them in whole-group activities to make the vocabulary meaningful to them, and practiced the skill of breaking down a problem and creating steps to solve it. Table 13 shows the number of times that an element of CT was referenced in a lesson. The symbols in the table indicate if the teacher provided definitions of the element and if the students engaged in an activity related to that element. An example of student engagement from Ms. Beck's class was asking the students to discuss with a partner when they had created a list of steps (algorithms) or broken down a problem to make it easier to solve (decomposition).

Finding Three: Teachers in RCSD did not have a common, shared understanding of the meaning of CT as defined by the elements identified by Angeli et al. (2016) or otherwise. Teachers suffered from definitional confusion related to CT and struggled to make sense of their own interpretations of CT, even when provided with concrete definitions and relevant examples.

The third finding of this capstone study reviews the ways in which teachers under investigation defined computational thinking and how that definition compared to the elements of CT identified by Angeli et al. (2016). Based on the field notes and transcripts

from my observations and interviews, I found that the teachers primarily confused CT with general technology use or how well someone is able to perform certain tasks on a computer. The following section describes how each teacher defined CT in their interviews and teaching.

Susie Jones

In her first lesson, Ms. Jones planned to teach computational thinking explicitly, and she defined it for her students as "a way to understand and solve problems by breaking them down and creating steps to solve them" (Jones, first interview). Her second lesson did not target CT, and instead focused on the importance of and ways of creating strong passwords. During the second interview with Ms. Jones, which came two weeks after her first observation, Ms. Jones defined CT as "the way you use technology to help you do your thinking like a computer" (Jones, second interview). This definition differs from that which she provided to her students during her first lesson and requires the use of technology for computational thinking.

During her second interview, I gave Ms. Jones a copy of the elements of CT identified by Angeli et al. (2016) and examples of each element as could apply to instruction in third grade (Table 1, p.2). After reviewing the table, Ms. Jones provided general examples of the ways that her students learn and practice skills of CT that did not differ from the examples in Table 1. For example, she said that they learned generalization "in math with different patterns," and abstraction in "a lot of different ways, even with reading. Also, in math, with geometry and 2D/3D shapes" (Jones, second interview). These examples are very similar to the examples provided in Table 1.

During her second interview, Ms. Jones did not offer examples of the ways that her students learned and practiced CT skills besides those already given to her.

The data collected suggest that Ms. Jones was unsure about computational thinking. Her two lessons each demonstrated a different understanding of CT, and the definition that she gave in her interview shows further variance. Her understanding of CT sometimes included basic technology use.

Sophie Horton

Ms. Horton demonstrated an understanding of CT that equated it to general technology integration into instruction. For example, when asked what CT she wanted me to see in her first lesson, she said that she "planned for [me] to see how the kids access their documents on Google Drive… inserting pictures and adding captions" (Horton, first interview). To her, students were engaged in CT by using their laptops and putting their notes into a digital format. This relates to basic technology use.

When asked how she defined CT during the second interview, Ms. Horton's response varied across two different themes within technology use. The first theme was on how technology can enhance learning; "whoever is using the technology, just how they are using it to enhance their… goal. And how… they are using it in a way that's different" (Horton, second interview) than traditional paper-and-pencil schoolwork. Troubleshooting is another theme of CT for Ms. Horton, specifically how the users of the technology:

> Solve the small problems that arise from using technology along the way… and what skills do they need in order to problem-solve something that they stumble

> upon while they're working, and can they remember to click the undo button if they've accidentally deleted (Horton, second interview).

Ms. Horton's understanding of CT related to these two themes, technology to enhance learning and troubleshooting, was evident in the lessons that she chose to teach, during which she sought to enhance students' learning experiences by moving them to a digital, collaborative workspace such as Google Slides and Sheets.

When asked during her second interview to review the elements of CT and examples (Table 1, p.2) and to provide examples of the ways that her students learn and practice skills of CT, Ms. Horton's answers focused primarily around her class's focus on Maker Education. Maker Education encourages teachers to promote student awareness of the way the world around them is designed, and promotes the idea of tweaking, tinkering, modifying or hacking that design to improve it (MakerEd, 2019). In the past, teachers in RCSD have been encouraged to integrate Maker Education into their classrooms. Ms. Horton said that generalization, decomposition, and algorithms are each a part of their Maker curriculum. Ms. Horton did not describe how any of these elements of CT related to the use of technology despite her own provided definitions of CT that involved technology use.

Carmen Sanchez

Ms. Sanchez's understanding of CT demonstrated through her first lesson observation and interview was in line with that of her co-teacher: CT is equivalent to general technology use and the creation of something using technology. Ms. Sanchez deviated from her co-teacher for her second lesson and the responses during Ms. Sanchez's interview showed a shift in her thinking from her first lesson and interview.

When asked how the lesson specifically addressed CT, Ms. Sanchez referenced a learning standard. She said the lesson addressed CT in the way that it "hit the specific standards, talking about having kids think like a computer, following steps like a computer would and coding. In general, having kids work through a sequence of events to make a certain action happen on their computer" (Sanchez, second interview). To her the students were engaged in CT by thinking like computers:

> They had to think like a computer, so they had to make sure that all their code was lined up correctly. Otherwise, their actions wouldn't happen. So yeah, so they just had to think about the commands exactly that they needed to use in order to make what they wanted to happen, happen (Sanchez, second interview).

This idea of CT as thinking like a computer differs from her original opinion that she previously shared during her first interview, that CT was related to general technology use and the creation of digital artifacts.

Ms. Sanchez continued to refer to an unspecified learning standard. While still describing the ways that she believed her students were engaged in CT throughout the lesson, Ms. Sanchez offered her own definition of CT:

> Yeah, so I guess what I'm understanding, at least from the standards, of what computational thinking is, would be where kids are, I guess, from what the standards are saying, is it's where you're thinking or acting like a computer would, by following steps and creating sequences of events that would happen based on your actions. And that's what they were doing in order to create this game. In order to be able to create a product that they wanted to do (Sanchez, second interview).

Something about Ms. Sanchez's understanding about CT had changed between her two observations, but she was not without some level of uncertainty between her prior convictions and her newly evolved understanding. When asked why she chose to teach this lesson, she said that "this lesson really [did] embody what computational thinking is, of thinking like a computer, of creating something really cool with a computer" (Sanchez, second interview). She maintains the idea that CT involves the creation of something through technology use, even as she begins to see it as something more.

When asked to define computational thinking, Ms. Sanchez referenced standards again, but this time, the Virginia SOLs specifically. She indicated that she tried to look up the definition of CT in the third grade SOLs but could not find it:

> Computational thinking is not defined in the 3rd grade Standards of Learning! You have to go to the one that's K-12 and then they have this brief little message about what computational thinking is. So I did a little research online, just to kind of clarify if I knew what it meant. But I kind of feel like it has something to do with having kids think like a computer (Sanchez, second interview)

She was unable to find a definition of CT when looking at the state-provided resource for her grade level and had to look elsewhere. The use of resources will be reviewed below in the discussion of the fourth finding.

Ms. Sanchez was aware of her own definitional confusion about CT, recognizing that her prior assumptions about the topic may not have been accurate, but she still was not sure. While still struggling to define CT in the interview, she wondered if her "understanding is correct, and [she's] not sure if it is honestly, it's just literally from what [she] read, that computational thinking is different, different than, in general, just

working on the computer" (Sanchez, second interview). Even after reading a definition of CT from the Virginia SOLs for Computer Science and doing her own research online, she *still* was not sure if CT was more than *just working on the computer*. Her prior conceptions were so strong that even evidence to the contrary was not enough for her to overcome them.

Ms. Sanchez said that her prior understanding of CT was having students "creating artifacts or projects, you know, having kids be able to know how to turn a computer on and problem solve [re: troubleshoot] when your internet's not working or being good digital citizens. Things like that" (Sanchez, second interview). At this point in the interview, she claimed that she no longer held that belief, instead moving onto the idea that CT is thinking like a computer. Earlier still, she held onto another idea, that computational thinking relates to addition and subtraction. Ultimately, prior knowledge of my previous position in RCSD as an ITRT influenced Ms. Sanchez to believe that I was not "writing [my] thesis on addition and subtraction" (Sanchez, second interview). That prior knowledge of my profession led both Ms. Sanchez and her co-teacher, Ms. Horton, to the conclusion that "maybe it's something related to computers" (Sanchez, second interview). This hints at the ways that both teachers at RFES may have developed their ideas about CT.

Ms. Sanchez's understanding of CT demonstrates tremendous ambiguity. She admits that her thinking has changed but is unable to let go of some of her earlier beliefs. After she gave her updated definition of CT that involved students thinking like a computer and following steps to create something using technology, I provided her with Table 1. I asked Ms. Sanchez to share examples of the ways that her students learn and

practice the skills of CT, and the few illustrations that she provided did not align with any of her previously stated notions of CT. An example of the ways that her students have used algorithms was through functional print writing, which she described as very similar to the example provided on Table 1. She said that her students practiced abstraction by paraphrasing or putting something into their own words, which was exactly the skill that students worked on in her first lesson, albeit done digitally with Google Slides. She offered no examples of generalization or decomposition and said that they had not done any debugging so far this year. Ms. Sanchez offered no examples of students doing anything to think like a computer, learning digital competencies, or any of her other ideas about CT.

Julia Beck

Ms. Beck introduced CT vocabulary words to her students during the first lesson first through an image (see Figure 11) and then again with her own drawing on chart paper (see Figure 12). The figures provided two different definitions for each vocabulary term, but it is fair to say that the definitions that she wrote down on Figure 12 were the definitions that she intended for her students to learn as she wrote them in her own kid-friendly terms based off her knowledge of her students.

Ms. Beck defined computational thinking through the terms decompose, algorithm, pattern matching, and abstraction. These elements are similar in many ways to the elements identified by Angeli et al. (2016). Table 14 contains a summary of the differences in vocabulary. She defined decompose as “break down a problem into smaller pieces” (Beck, second interview). Her definition of algorithms is “a list of steps that you can follow to finish a task.” Pattern matching is “finding similarities between things,” and

abstraction is “pulling out specific differences to make one solution work for multiple problems” (Beck, second interview). Ms. Beck did not offer her students a definition of computational thinking in any way other than through teaching this vocabulary, and likewise, when asked for her definition of computational thinking during her second interview, she said that she “think[s] computational thinking is like a summation of all of those four key vocabulary words” (Beck, second interview).

Table 14

Summary of vocabulary differences between Angeli et al. (2016) and Ms. Beck

Angeli et. al. (2016) definition	CT Vocabulary	Ms. Beck’s definition
The skill to decide what information about an entity/object to keep and what to ignore.	Abstraction	Pulling out specific differences to make one solution work for multiple problems.
The skill to formulate a solution in generic terms so that it can be applied to different problems.	Generalization/Pattern Matching	Finding similarities between things.
The skill to break a complex problem into smaller parts that are easier to understand and solve.	Decomposition	Break a problem down into smaller pieces.
The skill to devise a step-by-step set of operations/actions of how to go about solving a problem.	Algorithms	A list of steps that you can follow to finish a task.
The skill to identify, remove, and fill errors.	Debugging	N/A

While Ms. Beck maintained a consistent understanding and definition of CT throughout her two lessons and interviews, she still demonstrated some confusion during her

interviews. In one email that she sent to me while scheduling her observations, she indicated that her lessons would be utilizing a website called Flipgrid, which promotes video-based discussions from students through teacher-created prompts (https://www.commonsense.org/education/website/flipgrid). During her second interview, I asked her what caused her to deviate from that plan. She said that Flipgrid "wasn't a good representation of computational thinking in that, yeah, it was using technology, but it wasn't incorporating any of the like, main components of computational thinking… it is just using technology" (Beck, second interview). Here, Ms. Beck recognizes that this particular use of technology did not represent CT. However, later, when defining CT in her own words, she says that having the skills she has identified as computational thinking (decompose, algorithm, abstraction, and pattern matching), "you'll be successful in working with technology and like, the layers of technology, not only in coding, but in other programs you use with technology" (Beck, second interview). This suggests a core belief that CT has a direct relationship to supporting proficient technology use, similar to the misconception of CT being digital competencies. Interestingly enough, like Ms. Jones at HES, Ms. Beck asserts a relationship between CT and technology use, but does not use technology in her lessons that integrate CT.

The examples of the ways that her students learn and practice the elements of CT as described in Table 1 do not include technology use either. Ms. Beck' says her students use abstraction in their word study and when editing their writing. Generalization is done as students are "finding out how to solve things, like, generally" (Beck, second interview), such as in science, language arts or social studies. Decomposition is done as

students break down word problems in math, algorithms are used when doing “How to” writing, and debugging is done when they remove extraneous information from math word problems. Ms. Beck was the only participant in this capstone project who offered examples for the ways that she teaches all five elements of CT from Angeli et al. (2016). She also thinks that teachers are “teaching [CT] almost every day without even realizing it” (Beck, second interview). While the examples that she provided may not have aligned perfectly with definitions available to her in Table 1, it was promising that she recognized that CT could be integrated into and was relevant in most content areas that she taught.

Summary of Third Finding

The teacher participants in this study provided their own definition of CT and described the ways in which their students engaged in CT during the two lessons that I observed as well as in other lessons throughout the year. The definitions and examples from each teacher varied, but demonstrated that the teachers did have a common understanding of CT, nor were their individual definitions consistent with the literature on CT. There were similarities in the teachers’ responses that suggest that each of them think CT relates to how well students can perform certain tasks using technology, or digital competencies.

Examples of digital competencies can range from something as simple as knowing the difference between right and left clicking on a desktop icon or navigating through a file browser to find a file that was saved to more complex skills such as setting up a collaborative digital workspace for use of asynchronous and synchronous tools for collaboration. Within this misconception of digital competencies, teachers further distinguished CT across creation through the use of the use of technology, technology to

enhance an experience, troubleshooting, and basic technology use. Each of these themes is related to the observations and interviews with the four participant teachers.

Finding Four: The lessons that touched upon elements of CT were taught by two teachers who used CT resources. Resources include lesson plans, articles, graphics, and instructional videos. Two teachers who did not touch upon elements of CT did not use CT resources.

In determining the fourth finding of this capstone, I examined the ways that teachers described their lesson planning in advance of my observations. In the section below, I identify and review any relevant resources that teachers used when planning for the lessons in which they sought to teach CT. I organize these resources by each teacher, as there were several resources used by multiple teachers, albeit used in different ways. I found that only two teachers used explicit CT resources when planning for their lessons taught as a part of this capstone study. Those resources include lesson plans, articles, graphics, and instructional videos. A summary of the explicit CT resources used by teachers while planning is available in Table 15.

Table 15

Summary of Resources used by teachers while planning

Resource Title	CT Elements	Type	Source	Lesson	Used?
Susie Jones					
2017 Computer Science Standards of Learning - Grade 3 (Appendix F)	Ab, G, Dc, Al, Db	SD	Teammate	Both	Yes
BrainPop Jr. - Computational Thinking	Dc, Db	VS	Self	1	Yes
Computational Thinking Lesson Plan - Decompose!	Dc, Al	LP	Self	1	No
2013 Computer Technology Standards of Learning for Virginia Public Schools - Grades 3-5 (see Appendix M)	None	SD	Teammate	Both	1 – No 2 – Yes
Plan Engaging STEM Lessons: STEM Lesson Plan Template	None	LPR	Self	1	No
Lesson 3: Debugging in Maze (Code.org)	Db	LP	Self	1	No
Code.org Unplugged Computational Thinking (Appendix M)	Ab, G, Dc, Al	LP	Self	1	No
Strong Passwords (Appendix K)	None	LP	Self	2	Yes
Sophie Horton					
Computational Thinking (Figure 11)	Ab, G, Dc, Al	G	Self	N/A	N/A
What is Computational Thinking?	Ab, G, Dc, Al	G	Self	N/A	N/A

Carmen Sanchez					
Programming for Kids and Everybody: Learn Scratch Programming	None	Y	Self	2	No
Scratch Tutorial I: Make Your First Program	None	Y	Self	2	No
Scratch Coding: A complete overview for beginners	None	Y	Self	2	No
2017 Computer Science Standards of Learning - Grade 3 (Appendix F)	Ab, G, Dc, Al, Db	SD	Self	2	Yes
2017 Computer Science Standards of Learning - K-12	Ab, Dc, Al	SD	Self	2	Yes
Coding Games in Scratch - How to build Star Hunter (Appendix L)	None	H	Self	2	Yes
2013 Computer Technology Standards of Learning for Virginia Public Schools - Grades 3-5 (Appendix N)	None	SD	Self	2	Yes
Julia Beck					
2017 Computer Science Standards of Learning - Grade 3 (Appendix F)	Ab, G, Dc, Al, Db	SD	Self	Both	Yes
2013 Computer Technology Standards of Learning for Virginia Public Schools - Grades 3-5 (Appendix N)	None	SD	Self	Both	No
Computational Thinking (Figure 11)	Ab, G, Dc, Al	G	Self	Both	Yes
Code.org Unplugged Computational Thinking (Appendix M)	Ab, G, Dc, Al	LP	ITRT	1	Yes
Computational Thinking for Kids	None	A	Self	Both	No
What is Computational Thinking?	None	A	Self	Both	No

Create-Your-Own-Stories ROUGH DRAFT (Appendix O)	Ab, G	LA	ITRT	2	Yes

Note. Ab = Abstraction; G = Generalization; Dc = Decomposition; Al = Algorithms; Db = Debugging; SD = Standards Document; LP = Lesson Plan; LPR = Lesson Planning Resource; VS = Video for Students; G = Graphic; Y = YouTube video; H = Handbook; A = Article.

Susie Jones

For the first lesson that Ms. Jones taught on CT and decomposition, she began her planning with the Virginia SOLs for Computer Science for third grade. While reviewing the standards, she kept her students in mind, saying "it was hard to figure out exactly how this could relate to these, the third grade class that we have" (Jones, first interview). She ultimately landed on SOL 3.6, "The student will break down (decompose) a larger problem into smaller sub-problems, independently or collaboratively" (VDOE, 2017). This standard comes under the subheading "Algorithms and Programming." After deciding on the SOL she wanted to teach, Ms. Jones conducted a web-search for resources on teaching computational thinking for "about an hour." She "looked online for different things," "different ideas of computational thinking lessons," because she "wanted to know what was out there for computational thinking" (Jones, first interview).

In her web-search, Ms. Jones arrived at several different resources that she used to develop her content knowledge for CT. She "got a lot from BrainPop Jr." (Jones, first interview) and then found a number of different websites or articles. I will describe the articles in order of relevance to the lesson that she taught, from least relevant to most relevant.

No computational thinking. The least relevant resource that Ms. Jones accessed while planning her first lesson was a lesson planning resource called "Plan Engaging STEM Lessons: STEM Lesson Plan Template." This resource does not relate to CT, and use was not evident in her first lesson. Another resource that Ms. Jones accessed that was not particularly relevant to her lesson planning was the 2013 Computer Technology Standards of Learning for Virginia Public Schools – Grades 3-5. This standards

document does not reference or allude to any CT or its component parts, and Ms. Jones recognized this while planning. Ms. Jones did not mention this resource while discussing her second lesson, but her second lesson does address SOL 3.5.3B. The standard states that the student must "Discuss and model responsible behaviors when using information and technology; identify reasons for taking security precautions when using any technology, especially those related to the Internet; and demonstrate responsible behavior, such as using strong passwords and avoiding high-risk activities" (VDOE, 2013).

Computational thinking, not used. Ms. Jones accessed two resources from Code.org that directly relate to CT or the elements identified by Angeli et al. (2016), but use of these resources in her lesson was not evident, although they may have influenced her understanding of CT. The first was a lesson from Code.org's CS Fundamentals 2018 Course C, Lesson 3: Debugging in Maze. This lesson plan makes no mention of CT, but does define debugging. Ms. Jones did not reference debugging during her lesson. The second resource, an unplugged lesson plan on computational thinking, defines its own four components of CT. I review this resource, which is found in Appendix M, later in this chapter.

Primary resources. The two resources that had the greatest observable influence on Ms. Jones were from BrainPop Jr., as she indicated during her interview. A lesson plan from BrainPop Jr. called "Computational Thinking Lesson Plan: Decompose!" was the primary instructional resource used in planning for Ms. Jones's first lesson, and Ms. Jones provided me with a print copy of the lesson plan with her own annotations. The lesson plan included a definition of CT, which is the definition Ms. Jones used in her first

lesson and gave in her first interview. Ms. Jones followed this lesson plan closely, including accessing the instructional video for her students during the lesson. The lesson plan included a link to background information on computational thinking, but that link was broken on the BrainPop website.

In order to plan for her second lesson in which she intended to integrate computational thinking, Ms. Jones again began with the VA SOLs for Computer Science. This time, she selected SOL 3.11, "The student will create examples of strong passwords, explain why strong passwords should be used, and demonstrate proper use and protection of personal passwords." This standard comes under the subheading "Cybersecurity." As described in sections above, Ms. Jones's second lesson did not include instruction on CT, either implicitly or explicitly. Her planning for this lesson is worth mentioning, however, because she returned to the Computer Science SOLs to identify her instructional objective. As with her first lesson, Ms. Jones searched the web to find ways to teach the new standards. Her search yielded a lesson from Common Sense Media for grades 3-5 called "Strong Passwords." The learning objectives for the lesson from Common Sense Media were as follows: "Students will be able to identify the characteristics of strong passwords; apply characteristics of strong passwords to create new passwords; and create secure passwords with their family members" (2017). As was the case with her first lesson, Ms. Jones spent about an hour researching and planning, and primarily followed the lesson plan she found on the Internet while teaching.

In the next two sections, I explore the ways the teacher participants at Rusty Falls Elementary School planned to integrate CT into their instructional lessons.

Sophie Horton

Sophie Horton taught two lessons that did not teach CT, either explicitly or implicitly, as it is defined in this capstone project. Ms. Horton taught lessons that showed general technology integration into content and gave students the opportunity to use technology to create something new. The resources that Ms. Horton utilized were only content-area resources, such as a book about Ancient Egypt.

I requested a third interview with Ms. Horton in order to clarify some of the concerns she had discussed during her previous interviews about student technology use at RFES. Ms. Horton revealed that after the consent agreement meeting with the third grade team at RFES, Ms. Sanchez presented a Standards of Learning document that she had not seen before to her while co-planning after school. She did not say whether these were the SOLs for Computer Science or Computer Technology. When Ms. Sanchez showed them to her, she admitted that she did not review them thoroughly:

> I looked at them, but I didn't look at them in depth because it looked like a very long document. So I looked at them, but not, not directly, point-to-point. So I kind of just gathered from it, truthfully, I gathered from it, 'Oh, it has to do with technology and using computers, my kids, and getting them to problem solve.' So I was like, 'Okay, yeah. I know that kind of what you [referring to me], your work is.' And I thought, 'Okay, I'll just use technology with my kids and we'll do some projects with that and along the way they'll solve problems within whatever curriculum or teaching.' (Horton, third interview)

Her statement led me to believe that she was referring to the 2013 Computer Technology Standards of Learning, because of her reference to "my work" as a former ITRT, and to

problem solving, knowing that she views problem solving as similar to technology troubleshooting. However, regardless of which standards document it was that she saw, she admits that she did not integrate those standards into her lesson while planning, and taught to her own interpretation of the SOLs, but not the specific standards themselves.

The day after the second interview, Ms. Horton searched for "what is computational thinking?" through Google's image search (Horton, third interview). The image that she clicked on, from ComputationalThinkers.com (see Figure 11) "really surprised her." Because of the search, she said CT "didn't seem like it was just about technology but it was more about kids problem solving in a way that a computer might. So like, using… computer type ways of solving their problems whether it's with technology or not" (Horton, third interview). The resource may have helped her own understanding of CT, but since she accessed it after the second observation and interview, her improved understanding had no impact on her current students, but might on future students.

Carmen Sanchez

Like Ms. Horton, the first lesson that Ms. Sanchez taught did not integrate CT nor did it teach it explicitly. Ms. Sanchez planned her first lesson with Ms. Horton, utilizing books and YouTube videos about Egypt, and the content standards for Language Arts and Social Studies.

For her second lesson, Ms. Sanchez diverged from the path of her co-teacher. Rather than teaching a lesson on graphing using Google Sheets, Ms. Sanchez gave her students the opportunity to create a game using Scratch. To plan for the lesson, Ms. Sanchez spent thirty minutes one afternoon at school reading through a book she

purchased, “Coding Games in Scratch” by John Woodcock (2016) and then another half an hour on another day watching several videos from YouTube “of people working in Scratch” (Sanchez, second interview). She also visited the Virginia Department of Education’s website to look through the Standards of Learning that reference computational thinking. I provide details about these resources below.

Scratch handbook. The book introduces coding and programming in general, but primarily focuses on teaching the Scratch interface and layout through instructions on how to make four increasingly complex games. While there is no reference to CT or to the elements identified by Angeli et al. (2016), there are instances of algorithms, debugging, and CT through the aspects of writing code. Woodcock’s guide provides exposure to coding in Scratch through simplified instructions, but it does not provide the users with the opportunities to choose their direction with minor exceptions of “hacks and tweaks” at the end of each set of instructions. Ms. Sanchez used the guide as it was intended, as a step-by-step set of instructions for the students to create the game.

Tutorial videos. Ms. Sanchez watched three different YouTube videos in an effort to enhance her knowledge of Scratch. The first video, “Scratch Tutorial 1: Make Your First Program,” was published by Kevin Briggs on December 28, 2015. This video focused on the layout of the program and how to use one set of scripts to cause motion in the sprite. While this video highlights some of the scripts, it did not teach the audience how to use Scratch. The second video, “Scratch Coding: A complete overview for beginners,” by Flipped Classroom Tutorials, was published on March 20, 2018. This video starts by showing the audience how to visit the Scratch website, create an account, and become a member of the Scratch open source community. This information is not

relevant to users in RCSD, as Scratch is installed on their computer and students under 18 years old need parental permission to create accounts. The video targeted teachers who wanted to teach their students Scratch, at one point saying to "remind your students" (Flipped Classroom Tutorials, 2018) about a certain element of the program. The production quality of the video was high, and featured a table of contents of the instruction within the video description, and the author responded to questions in the comments section on YouTube. The third video that Ms. Sanchez sampled was published by Ameer Fazal on April 9, 2017, and is titled "Programming for Kids and Everybody: Learn Scratch Programming." This short video shows how to download Scratch to your computer and introduces ways to cause the sprite to move, make sounds, and turn. The video is too fast and lacks the production quality of the second video. There is no real description on how to use Scratch, as the author instructs the audience to "just dive right in" (Fazal, 2017). None of these videos refer to CT or the elements of CT identified by Angeli et al. (2016). The second video, by Flipped Classroom Tutorials, is the only one that does much to teach how to use Scratch. Based off of the instruction that I observed from Ms. Sanchez during her second lesson, it is not clear whether or not any of these videos had any influence on her ability to use Scratch, as she simply followed along with her students through "How to build Star Hunter" in Woodcock's (2016) guide.

State standards documents. In addition to the Scratch guide and YouTube videos described above, Ms. Sanchez also utilized the Virginia Department of Education website on Standards of Learning (SOL) and Testing (http://www.doe.virginia.gov/testing/index.shtml). This site contains links to all of the different documents for content areas with SOLs, such as English, Mathematics, and Computer Science. Based off the

description that Ms. Sanchez provided, it was unclear whether she was referring to the standards for Computer Science or Computer Technology:

> So we found this whole set of standards, except when I looked at the third grade standards, there was nothing related to computational; it divided it into, I can't remember the term, but something. I think they just, I don't know what they called it, but in general something just, it just seemed like there were standards on how to use a computer, how to be a good digital citizen, how to create artifacts on a computer. (Sanchez, second interview).

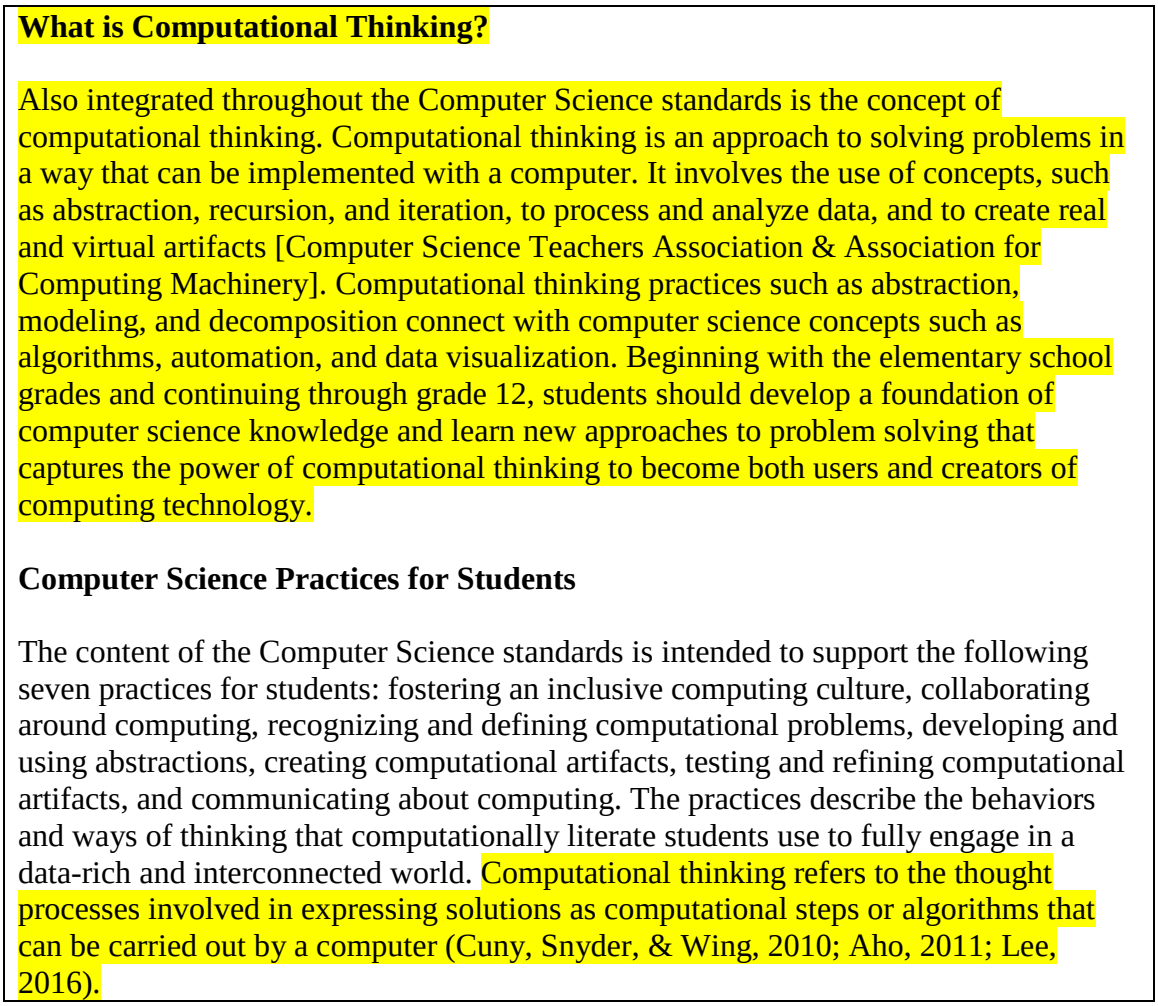
This could refer to either set of standards, as both have instances of basic computer use, digital citizenship, and creation.

Ms. Sanchez's confusion is warranted, as neither the standards for Computer Science or Computer Technology reference CT, so she kept searching:

> On the third grade standards, they didn't break anything out to say 'computational thinking,' so I like, opened up the K-12 [All Computer Science Standards of Learning Document] and they have this brief little paragraph about what is computational thinking. And my understanding of it, because in my mind, it's not, I don't think it was a great description for someone who like, I mean, I know my husband codes but I don't necessarily, and you know, I know that I'm just learning how to code a little bit on Scratch, you know. But I don't feel like it was very clear, exactly what is, what they're looking for. (Sanchez, second interview).

Ms. Sanchez provided me with the copy of the K-12 document that she referenced above (See Figure 13), and it is confusing. There is a definition of CT under the header "What is Computational Thinking," but there is another definition in the section below called

"Computer Science Practices for Students." While neither definition is directly contradictory to the other, to a novice without content knowledge about CT like Ms. Sanchez, having two different definitions can be confusing.



What is Computational Thinking?

Also integrated throughout the Computer Science standards is the concept of computational thinking. Computational thinking is an approach to solving problems in a way that can be implemented with a computer. It involves the use of concepts, such as abstraction, recursion, and iteration, to process and analyze data, and to create real and virtual artifacts [Computer Science Teachers Association & Association for Computing Machinery]. Computational thinking practices such as abstraction, modeling, and decomposition connect with computer science concepts such as algorithms, automation, and data visualization. Beginning with the elementary school grades and continuing through grade 12, students should develop a foundation of computer science knowledge and learn new approaches to problem solving that captures the power of computational thinking to become both users and creators of computing technology.

Computer Science Practices for Students

The content of the Computer Science standards is intended to support the following seven practices for students: fostering an inclusive computing culture, collaborating around computing, recognizing and defining computational problems, developing and using abstractions, creating computational artifacts, testing and refining computational artifacts, and communicating about computing. The practices describe the behaviors and ways of thinking that computationally literate students use to fully engage in a data-rich and interconnected world. Computational thinking refers to the thought processes involved in expressing solutions as computational steps or algorithms that can be carried out by a computer (Cuny, Snyder, & Wing, 2010; Aho, 2011; Lee, 2016).

Figure 13: Excerpt from Computer Science Standards of Learning for Virginia Public Schools (VDOE, 2017), highlighting by Carmen Sanchez.

The lack of clarity and simplicity in the standards documents frustrated Ms. Sanchez, and that was apparent during the interview. The two different definitions, combined with her own prior misconceptions about CT from her first interview, created further confusion for her about what she and her students were supposed to be doing. She wondered, "so then I'm like, are they just trying to, are they trying to apply what you do

when you're coding to other situations? And that, just again, like, I just felt like that was really unclear" (Sanchez, second interview). In her confusion, she chose to latch onto the concepts with which she was most familiar.

In the next section, I review the way that the final participant in this capstone study, Julia Beck, planned and prepared to integrate CT into her instructional lessons.

Julia Beck

Ms. Beck took a different approach to planning than did the other three participants in this capstone study in that she planned both lessons together. In her first lesson, she introduced students to CT vocabulary and taught them how to play the "Game with No Rules." In her second lesson, she reviewed the same CT vocabulary with students and then instructed them to play the "Game with No Rules" again, but this time using content-specific vocabulary, related to their language arts unit on Folk Tales.

To begin, Ms. Beck did some research on her own. She visited the VDOE website and found the Standards of Learning for Computer Science (see Appendix F) and the Computer Technology Standards of Learning for Virginia Public Schools (see Appendix N). She also did a web search of "computational thinking simple definitions" (Beck, first interview). Her web search led her to a graphic that formed the basis of her understanding of CT and the vocabulary that she introduced to her students (see Figure 11). She also accessed two written resources online that said they were about CT.

One resources that Ms. Beck accessed was a blog post from IMACS, the Institute for Mathematics & Computer Science. The post, titled "Computational Thinking for Kids," is for parents who want to make computer science relevant at home in an abstract way. It makes reference to Wing's (2006) work introducing CT, saying that it "clearly

explains what CT is and is not" (IMACS, 2012), but then goes on to simplify CT as abstract thinking. This resource does not reference any elements of CT, but instead provides parents with ways that they can introduce CS concepts like stacks and queues or object oriented programming to young children through unplugged activities.

The other resource that Ms. Beck shared with me was the article "What is computational thinking?" by Tim Slavin (2014). Slavin describes CT as a "structured way to solve problems" (2014), and makes reference to Wing (2006) as well. Slavin emulates Wing in the way that he offers non-specific examples of CT, calling it "fundamental, not a rote skill," and "a way that humans, not computers, think" (2014). Before the article goes into much more detail about CT besides reciting Wing, it requires payment. I did not ask Ms. Beck if she chose to pay for the article.

Computational Thinking Unplugged. The root of Ms. Beck's lesson was the Code.org unplugged lesson plan from their Code Studio, Course 3 (see Appendix M). Code.org recommends this lesson for students in grades 4-5 who have previously completed Courses 1 and 2. Depending on where Ms. Beck accessed the lesson plan, there may have been a 95-second video introduction to the lesson, "Unplugged – Computational Thinking," complete with an introduction of the target vocabulary and the activity students will complete. The intended audience of the video is the student. Below the video, there are two links to access the full lesson plan. The lesson organization is comprised of an overview, a teaching summary complete with links, and a teaching guide that includes a list of resources needed by the students and the teacher for the lesson. Resources for the teacher include a different video, "Course 3 – Computational Thinking," a seven-minute discussion from a Code.org instructor on how to teach the

lesson and anticipate student difficulties. The video was specifically created for this lesson. There are also copies of the resources the students will need, including the "Computational Thinking Kit," which Ms. Beck used with her students during her first lesson, and an assessment, which was not used in Ms. Beck's lesson.

The lesson plan includes the four vocabulary words that Ms. Beck introduced to her students: Decompose, pattern matching, abstraction, and algorithm, along with their associated definitions. It walks the teacher through the intended way to introduce these terms and definitions to the students. The next section of the lesson plan provides a detailed guide for teachers to follow to introduce the idea of CT to the students and have them practice using these four parts of CT with teacher-support. This part of the lesson plan is math intensive, and Ms. Beck ultimately decided that it was too complex for her students at this time, so she skipped it. Instead, Ms. Beck had her students focus on the vocabulary, and how to make it relevant.

The third section of the lesson plan was the activity. Ms. Beck's students worked on this activity during my first observation using the "Computational Thinking Kit" referenced above, and it was this kit that was modified to create the "Create-Your-Own-Stories ROUGH DRAFT" that Ms. Beck used in the second observation. The activities section tells teachers what to say to students in order for them to play the "Game with No Instruction" [the "Game with no rules" in Ms. Beck's lesson]. The lesson plan concludes with a student wrap-up, assessment, and connection to national-level standards from organizations such as ISTE, CSTA, and NGSS. Ms. Beck did complete a wrap-up activity, but it differed from the one suggested in the lesson plan, and her students did not complete the assessment. It is unclear as to whether or not she planned for them to

complete the assessment, but most student groups were unable to complete the "Computational Thinking Kit" in the allotted time, so it would not have been appropriate to give them the assessment.

While the lesson was designed to be complex, Ms. Beck recognized that her students were not prepared to complete the whole lesson. It is recommended for older students, and was designed to be completed as a part of a sequence that students at RES did not complete. The part of the lesson that Ms. Beck kept, the "Game with no Instructions," was modified for integration with content- specific lessons. I will review this resource below.

Modified resource. The Code.org "Computational Thinking Kit" was modified so that Ms. Beck could use the same type of activity with her students, but with content-area vocabulary integrated into the activity. The modified resource is the "Create-Your-Own-Stories ROUGH DRAFT" (see Appendix O.) Where the original activity provided students with a list of colors, items (Ex: cell phone, pineapple, cupcake), body parts, and animals that they would select by rolling a single die. The original activity provided these choices to the students. The modified activity tasks students with coming up with their own list of six choices for the following categories: characters, setting, problem, solution, and extra, which is optional. That is the primary modification.

The remainder of the instructions on the activity are identical, except that there are no examples provided for the student where they are supposed to "figure out how to play this game." So it is no longer a "game with no rules," but instead a game where students are supposed to remember the rules from use to use. As on the Code.org version, it instructs students to "use pattern matching and abstraction" to create a template to play

the game. Those instructions are the only instance of explicit CT integration in this modified resource, although it is clear that students are supposed to engage in the same type of thinking as on the Code.org activity, in addition to creating their own lists of characters, settings, etc.

Ms. Beck spent over an hour and a half planning and preparing for her two lessons that integrated CT into her instructional lessons. Still, at the end of each lesson, Ms. Beck wished she had more time to "delve deeper and do more" (Beck, first interview) with her students to ensure that they understood the CT she'd tried to teach. She was satisfied with her planning time, but wanted more, "always more time" (Beck, first interview).

Summary of Finding Four

In the sections above, I described the ways that the teacher participants in this capstone study prepared to integrate computational thinking into their instructional lessons. Some of the resources that teachers accessed while planning did not relate to CT. Many of the CT related resources that teachers looked for were to build their own understanding of CT. I reviewed the extent to which teachers made use of available computational thinking resources planning to integrate CT into their lessons. Some of the resources that teachers used provided them with a specific plan that they could use in their classroom, albeit modified to meet the needs of their students. Analysis of their planning practices show that the teachers who made use of available CT-related resources integrated elements of CT into their instructional lessons.

Finding Five: One teacher who taught lessons that touched upon elements of CT used district support personnel. Those personnel accessed and modified CT resources and co-planned the lessons with the teacher.

In the fifth and final finding of this capstone study, I identify and review any relevant support personnel from their school or district that teachers used when planning for the lessons that I observed. Only one teacher who participated in this capstone study sought the assistance from district or school support personnel.

To plan for her lessons, Julia Beck reviewed Virginia Standards of Learning for Computer Science and Computer Technology, found an image on the Internet with CT vocabulary, and read one and a half articles about CT. She took thirty minutes herself to research CT from the VDOE and the Internet. However, after doing her own research, Ms. Beck admitted that she was still struggling to come up with two lesson plans:

> I had done my own research on computational thinking but I wasn't wrapping my head around how to incorporate it in the classroom, like how I had it was very segmented but those four, the like, four main components of computational thinking, kind of go hand-in-hand. So it's not like I could just teach like, decomposition one day and then this one day because then it'd be really abstract for the kids. And I feel like at this age, they're not very, they're like, becoming abstract. They're still very concrete in what they're learning (Beck, first interview).

She had ideas about what to do, but thought that she could use support. When she reached a point of frustration, she sought the assistance of district-level technology support, the ITRT assigned to work with teachers at RES. RES has two ITRT assigned to the building on a part-time basis, but both of them, Spencer Warner and Christie Ferguson, came in to assist Ms. Beck with planning. The three educators had two planning sessions, lasting 45 minutes and 30 minutes each, to figure out how to "incorporate more computer science

and technology in the classroom" (Beck, first interview) after Ms. Beck spent 30 minutes researching on her own.

It was Mr. Warner, Ms. Beck says, that brought their attention to Code.org. He found that Code.org had a lesson plan for teaching CT (see Appendix M), although there were parts of it that Ms. Beck did not think her students could handle yet:

> So Code.org had a whole other like section that goes along with [CT] that I thought my kids weren't developmentally ready for yet, which was just introducing, like introducing computational thinking with math. And I felt this [the lesson she ultimately chose] was very simple and my kids could get this. And that math portion, I could go back, like re-circle back to later in the year when they're more comfortable with certain math aspects (Beck, first interview).

Ms. Beck knew her students did not have the math background yet to be successful with the entire lesson plan available on Code.org, but now knew it was available when they were ready for it.

Ms. Beck, along with her two ITRTs, were able to modify activities from the Code.org lesson that she would use in her first lesson, so that it could fit with a variety of content areas. She was full of praise for her ITRT for the assistance that they provided her:

> To lesson plan it, Spencer came up with a template to then use it for any subject area. So I could use it again. Not the exact same lesson, obviously, but in the sense that they would create something to then explain their learning. So it was like an additional step that was really creative and thoughtful. And I'm not taking credit for that. That was all Spencer (Beck, first interview).

Mr. Warner modified the “Computational Thinking Kit” activity from the Code.org lesson plan to make it content-neutral, so that as the students developed the CT skills in the first lesson, they could continue to practice them through content integration throughout the year. This modification led to the creation of the “Create-Your-Own-Stories ROUGH DRAFT” (see Appendix O).

The remaining participants in this capstone study did not seek out assistance from district or school support personnel. Ms. Jones cited a lack of time as the primary reason for not doing so. The other two members of her team withdrew their consent to participate in this capstone study during scheduling, so she was the only teacher on her team attempting to integrate CT into her lesson plans at this time. She did receive assistance from one of her teammates, however, during the consent agreement meeting. After the three teachers learned about the new SOLs in computer science, one of the other two non-participating teachers printed out copies of the 2017 Computer Science SOLs and the 2013 Computer Technology SOLs and distributed the copies to the entire team. Ms. Jones’s teammate accessed the VDOE website, retrieved the appropriate standards, and gave them to her.

Ms. Horton planned both of her lessons with her partner teacher, Carmen Sanchez. For the first lesson that they taught, Ms. Horton credited the second grade teachers for assistance. Ancient Egypt used to be a second grade SOL, and the teachers shared resources with them on the topic.

Ms. Sanchez taught one lesson where students used Scratch. Unlike for her first lesson, Ms. Sanchez planned this lesson in isolation, without the consult of her co-teacher, Ms. Horton. The teachers still communicated about the objective of this lesson,

and Ms. Sanchez said that Ms. Horton "knew that I wanted to lead Scratch groups over the next couple of weeks. I don't really know if [she's] necessarily really interested in learning Scratch" (Sanchez, second interview). And while she and her ITRT, Catherine Day, have "chit-chatted some" (Sanchez, second interview) about Scratch, it was not in reference to any specific lesson, and in regards to CT, Ms. Sanchez stated that she "[hasn't] had anyone talk to us about it either" (Sanchez, second interview).

Participants as support personnel

In preparing for this capstone study, I planned to study how one team of third grade teachers at one school integrated CT into their classroom instruction. Research question 2a seeks to explore the ways that that team of teachers would access support personnel in their school and the district, and I hoped to learn how the teachers leaned on each other as a resource for lesson planning and preparation. Unfortunately, I was unable to find a full team of teachers at any one school to participate in this capstone study. Luckily, two teachers from RFES did participate, so I was able observe the way that those two teachers used each other as support.

As described in the third chapter, Sophie Horton and Carmen Sanchez are collaborative teaching partners in one classroom at Rusty Falls Elementary School. The two of them taught and planned together at school and outside of school. The first lesson that I observed for each teacher were the same lesson. After seeing Ms. Horton's second lesson, I expected Ms. Sanchez to teach the same thing. However, due to scheduling complications and inclement weather, my second observation with Ms. Sanchez fell more than a week after my second observation with Ms. Horton. During Ms. Horton's second

interview, I shared with her the elements of CT identified by Angeli et al. (2016) that form the conceptual framework for this capstone study.

After seeing Table 1, Ms. Horton researched CT as a way to combat the confusion that she was feeling. Ms. Horton admitted to me that she told Ms. Sanchez that she thought they were mistaken in their belief that CT is akin to technology integration, digital literacy, and technology troubleshooting, and that it was something much different. After telling Ms. Sanchez, Ms. Horton said of her partner-teacher:

> When she heard that, she was just like, well, she, I think, just dismissed it… So she didn't, it didn't seem like anything I said actually affected what her plan was for today. She seemed to keep the same mindset of like, 'Well that doesn't make sense to me. That's confusing.' So she went back to kind of like, her original belief of what that was. (Horton, third interview).

In Ms. Horton's eyes, Ms. Sanchez dismissed her discovery, and made no changes to her second lesson plan. Nevertheless, sometime after that conversation, Ms. Sanchez sent me an email (see Figure 14) with the Standards of Learning information for not only her second lesson scheduled for that afternoon, but also retroactively for the first lesson. During her first interview, when asked what resources she used to plan the first lesson, Ms. Sanchez said "Our content standards. Our language arts standards for reading and writing" (Sanchez, first interview), and ten days later, after discussing CT with her partner teacher, she has added standards 3.7, 3.9, and 3.13 from the Computer Science SOLs.

Interactions with other instructional personnel aided teachers as they prepared to integrate CT into their instructional lessons, intentionally or unintentionally. Without

asking for help, two teacher participants in this capstone study were given standards documents by a teammate. One participant had her perceptions of CT challenged by her partner-teacher. One participant sought out assistance from members of the RCPS instructional technology support group. The teacher requested assistance in planning her lessons that successfully taught two lessons that contained references to CT. Teachers who did not ask for or receive support from instructional support personnel did not integrate CT into the two lessons taught as a part of this capstone study.

Thursday, 11:40 AM [Date of second observation]
From: Carmen Sanchez

Hey Bert-

I shared with you the presentations that students were working on last time (on Google Drive). We are going to work on creating a game in Scratch this afternoon. So I'll just give you the manual I'm using today.

Standards for last week:
Algorithms and Programming 3.7
Computing systems 3.9
Data and Analysis: 3.13
Reading 3.4 e 3.6 (b,c,d,f)
Writing 3.7 (a) 3.8 (a, d, e, f) 3.9 (a, f, i, k)
Research 3.10 (b, c, e)
Social Studies 3.2

Standards for this week:
Algorithms and Programming 3.1 (a,b,c) 3.2 (a, b, c) 3.3

Figure 14: Email from Carmen Sanchez in advance of her second observation

Summary

There were five main findings of this capstone study. First, teachers in this study only touched upon elements of computational thinking in three out of eight lessons that I observed. Those three lessons collectively touched upon all five elements of CT to varying degrees. Second, in the lessons that did touch upon elements of CT, the

instructional strategies adopted by teachers were mainly direct, didactic instruction through vocabulary lessons or instructional videos or step-by-step instruction for students to construct sequences in block-based or unplugged activities. The third finding of this capstone study is that the teachers who participated lack a common understanding of the meaning of CT. The fourth finding is that the teachers who touched upon the elements of CT accessed available resources when planning for their lessons on CT, but these resources were largely used for their own research purposes and did not translate to the classroom. The fifth finding of this capstone study is that only one teacher utilized support personnel when planning for her lessons on CT, although one participant teacher did act in the capacity of support personnel for another teacher participant..

The findings of this capstone study are specific to the case study, but carry implications for each school, RCSD, and the VDOE as a whole. I discuss implications and recommendations to RCSD and the VDOE in Chapter Five.

Chapter Five

The findings of this capstone study address the ways that teachers integrate computational thinking or do not integrate computational thinking into their instructional lessons and the resources teachers used to plan for those lessons, which has implications for current work in the state of Virginia and in Rockview County School District (RCSD) (see Table 12). The problem of practice examined in this capstone study was: How can we help teachers implement new standards for computational thinking and infuse the instruction into their lessons? This capstone study used a single-case study approach to explore the ways that four third grade teachers in RCSD purportedly integrated computational thinking into their instructional lessons and the planning that went into that instruction. In this capstone study, I explored how teachers used instructional resources and district or school support personnel to plan their lessons, and the ways that elements of computational thinking were integrated into activities and tasks that were meant to engage students in computational thinking.

Summary of Findings

The five findings of this capstone study relate to teachers' integration of CT in their instructional lessons, their understanding of CT, and the way that they planned for their lessons that included CT. The first finding is that only three of eight lessons contained elements of CT identified by Angeli et al. (2016). All five elements were included to varying degrees. The second finding is that the instructional methods used by teachers in their lessons that included CT were teacher-centered and emphasized

vocabulary acquisition, but no actual CT. The third finding is that teachers lack a common understanding of the meaning of CT. The fourth finding is that only two of the four teachers accessed instructional resources related to CT while planning. Finally, the fifth finding of this capstone study is that only one teacher used district-level support personnel while planning for their CT lessons.

Discussion and Implications

The connection between the five findings of this capstone study rests of the teachers' understanding of and preparedness to teach computational thinking. Teachers are responsible for content mastery within their assigned grade level of instruction to ensure that all students achieve the minimum learning expectations set forth by the VDOE as the SOLs. Third grade teachers must be able to teach every third grade SOL across their assigned content areas, be it mathematics, social studies, or computer science. If teachers do not know or understand the required content, they are expected to develop that understanding, and the best ways to teach the content. The ways that teachers understood CT differed, which ultimately influenced the ways that teachers prepared and taught their lessons during this capstone study.

The third finding, that teachers suffer from definitional confusion related to CT, offers insight into the first finding, that most lessons did not touch upon elements of CT. During each of the three lessons that touched upon the elements of CT, the teacher correctly named and defined at least one element of CT present in the lesson. During their post-observation interview, the confirmed that they intentionally included those elements in their lesson. Susie Jones planned to teacher her students decomposition. She gave them the definition of the term, provided them opportunities to explain ways that they break

down problems to make them easier to solve, and after her lesson, she told me that was exactly what she intended to do. Julia Beck knew that she wanted to teach abstraction, decomposition, generalization (see Table 14), and algorithms. She introduced those vocabulary terms to her students.

In contrast, neither Sophie Horton nor Carmen Sanchez taught lessons that contained any elements of CT. During their interviews, when asked to describe the ways in which their students engaged in CT through the lessons I observed, their responses largely focused around creating digital artifacts and developing digital competencies related to the use of technology. Their definitions of CT did not match definitions reviewed in the second chapter of this capstone study; as a result, their lessons and explanations of those lessons are based on a misunderstanding of what CT is. Teachers cannot successfully integrate CT or the elements identified by Angeli et al. (2016) if they do not have a clear, comprehensive understanding of CT. As this capstone study seeks to learn how to help teachers integrate these new CT standards into instruction, discovering what teachers did when faced with new content to teach was critical. This information will lead to implications and recommendations later in this chapter.

The fourth and fifth findings involve the ways in which teachers prepared for their lessons that integrated CT. Only two of the teachers accessed CT-related resources and only one of those two teachers used district-level support personnel while planning. The teachers at RFES did not access CT resources, and did not understand what CT was, so they did not teach CT. Ms. Jones did access CT resources, but those resources may not have been enough to help her change her understanding of CT, so the instructional strategies that she used from the resources were limited to teacher-centered methods. Ms.

Beck, the one teacher who both accessed resources and used district-level support, was able to include elements of CT in both of her lessons. However, those resources were not appropriate for students without the necessary background knowledge, so while her activities could have been student-centered, she relied on teacher-centered vocabulary instruction to progress through the lesson.

Without a clear understanding of CT—what it is that they are supposed to teach, teachers cannot adequately plan to teach it. When teachers cannot adequately plan to teach something, the quality of the instruction is more likely to be poor. Understanding of the CT, and then understanding of the ways to teach CT, drive the way instruction is planned for and delivered. For example, a teacher must know that algorithms refers to students devising a step-by-step set of actions to go about solving a problem, rather than simply following steps, in order to plan an activity where students create those steps. Understanding the content is necessary for teachers to effectively plan to teach the content.

This capstone study sought to address the problem of practice related to helping teachers implement new standards for CT and infusing the instruction into their lessons. Next, I will address the implications that arise from findings related to this problem of practice.

Helping Teachers Develop Understanding of CT

As described above, each teacher who participated in this capstone study had a different understanding of CT, and some teachers demonstrated a different understanding between their own two lessons and interviews. The resources that teachers accessed to develop understanding of CT, in particularly the Computer Science Standards of

Learning, are insufficient to support teachers' understanding when used in isolation. Ms. Jones used the Third Grade CS SOLs to pick out her lesson topics; decomposition and strong passwords were both contained in the standards (Jones, first interview; Jones, second interview). I believe that Ms. Jones did not possess a clear enough understanding of computational thinking to select the appropriate, relevant standards within the Computer Science SOLs that related to CT. SOL 3.6 references decomposition and is found under the heading "Algorithms and Programming," but SOL 3.11 on password strength falls under "Cybersecurity." Both of these categories fall under the domain of "Computer Science," but "Cybersecurity" may not specifically relate to "Computational Thinking," while "Algorithms and Programming" does.

Ms. Sanchez accessed the VDOE website in order to learn more about CT. She found the definitions of CT on the site were difficult to locate and confusing. CT was not present in her grade-level standards, and was defined in two different ways on another document that she would not regularly access (see Figure 13). She suggested that the VDOE provide a clear definition of CT to teachers:

> It would be really helpful, I think for anybody, for [the VDOE] to be like, "Hey, computational thinking is specifically talking about debugging codes," or like talking about loops, making loops in actions in codes, or make, or sequencing events in coding. And they didn't say that. (Sanchez, second interview).

Despite having two distinct definitions from the VDOE itself, she is unable to articulate the meaning of CT. Ms. Sanchez thinks that teachers won't find out about the new CS standards or CT "unless [they] go to the VDOE site, which if [they] already have [their] pacing guides printed out, there's no reason necessarily to go there, then [they] would

never know" (Sanchez, second interview). The VDOE website is not one that Ms. Sanchez thinks teachers would frequently visit, and it did not help her to develop her understanding of CT. State agencies must ensure that provided resources are sufficiently clear about the content they expect teachers to integrate into their lessons. Reading a definition of CT will not be enough to help teachers to develop an understanding of CT, but the lack of a clear definition was a limiting factor for teachers in this capstone study.

Building on and distinguishing from digital competencies. Because all 3rd-12th grade students in RCSD have a laptop computer, the ITRT team has worked for years emphasizing the need for students to be digitally literate. While their work has not been widespread, this emphasis on digital competencies, creating digital artifacts, collaborating electronically, and enabling students to use technology to support their life-long learning goals may have created a blind spot for teachers who have embraced this movement. Teachers like Ms. Horton and Ms. Sanchez who have collaborated with ITRT on projects related to digital competencies curriculum and classroom technology integration might hear "computer science" or "computational thinking" and automatically relate it to the technology integration already present in their classroom. Teachers who have not worked with ITRT on digital literacy might make similar assumptions. Teachers may need help recognizing that developing students' CT skills is different than developing their digital literacy skills. That recognition can only occur once teachers have a solid understanding of the meaning of CT or clear examples of CT instruction so that they can develop an understanding of CT.

Teachers need professional development and educative resources that support teachers in order to understand CT. Professional development must provide clear,

illustrative examples for teachers to understand definitions of CT operationally or in practice. Chapter Two described resources from Google, ISTE and CSTA, among others, that can help to develop teacher understanding of CT. Resources such as these and others that focus on teacher knowledge, curriculum, and lesson plans are available to teachers, but they do not know where to look, or even that they need to look for them.

Helping Teachers Implement High Quality CT Instruction

Teacher development of understanding the best ways to teach any content is a complex process, with no one formula that fits for every person, context, or instructional setting (van Driel & Berry, 2012). And the process is not necessarily linear, where instruction of content directly follows from understanding of content (van Driel & Berry, 2012). Teachers need content, resources, examples of good and bad instruction, and opportunities for reflection to improve their understanding of ways to teach (Buchholz et al., 2013).

Once teachers have a better understanding on the meaning of CT, they must use best-practice instructional strategies to teach CT to their students. Activities and tasks should allow for student engagement in CT. The two teachers who referenced CT and touched upon the elements of CT in their instruction did so exclusively through direct, didactic instruction, and focused on teaching vocabulary. As discussed above, over-reliance on this particular instructional strategy may stem from a lack of in-depth understanding of CT or how to teach CT.

Accessing and implementing available CT resources. Teachers with understanding of a concept can design their own instructional experiences for students, or as Hattie (2009) suggests, they can find lessons that are already available online that are

designed to teach that concept. Ms. Beck made use of a resource that her ITRT found online for her. She recognized that one part of the activity that was especially math-intensive was beyond her students' current abilities. Her understanding of the third grade mathematics curriculum and knowledge about her specific students allowed her to make the instructional decision to exclude that part of the lesson when she taught. However, her lack of CT understanding or adequate planning kept her from recognizing that the activities in the resource were inappropriate for her class. Ms. Beck's first lesson included the use of the Computational Thinking Kit from Code.org (Appendix M). The purpose of the kit was to engage students in abstraction, decomposition, pattern matching and algorithms. In actuality, some students practiced abstraction and pattern matching through the kit, but most did not understand what they were supposed to be doing with the Computational Thinking Kit. None of the groups made enough progress through the kit to get to the points where they would be working at decomposition and algorithms. Her students had not completed the prerequisite courses on Code.org related to CT that provided background knowledge and were necessary to complete the activity. Ms. Jones and Ms. Beck both assessed the students on their abilities to recall definitions of the CT vocabulary introduced in class. Again the lack of familiarity with the attributes of CT made it difficult for the teachers to plan a CT lesson that was appropriate for their students.

Ms. Sanchez taught a lesson from a manual where students learned about the coding program Scratch, and how to use it on their computer. She chose this lesson because it represented her understanding of computational thinking, which meant students thinking like a computer and creating a digital artifact (Sanchez, second

interview). This introductory lesson on Scratch did not allow students the opportunity to engage in CT, but it may have been necessary to introduce the basics of computer programming, and might serve as a precursor to later CT engagement for her students. While there were no elements of CT explicitly named in this particular lesson, it does satisfy the requirements of SOL 3.2 in the Algorithms and Programming section of the Computer Science SOLs. SOL 3.2 states that:

> The student will construct programs to accomplish tasks as a means of creative expression using a block or text based programming language, both independently and collaboratively
>
> - using sequencing;
> - using loops (a wide variety of patterns such as repeating patterns or growing patterns); and
> - identifying events (VDOE, 2017).

This standard differs slightly from the algorithms as defined by this capstone study, where one must create the instructions, rather than simply follow them (Selby, 2014). Ms. Sanchez's lesson on Scratch may have introduced students to a programming language, but it was only the first step towards engaging them in the elements of CT identified by Angeli et al. (2016), each of which can be present while programming in Scratch.

It is important for teachers to have a plan for integrating CT and the new CS SOLs into their instruction. School districts in Virginia need a curriculum map that details the longitudinal plan for CT and CS integration. This can promote development of background knowledge, scaffolding, and vertical alignment across grade levels.

Team-based planning and collaboration. In order to building understanding of ways to teach, teachers must be able to work with other teachers to exchange ideas, strategies, and solutions to problems that arise while teaching (van Del et al., 1998).Ms. Beck and Ms. Jones were the only teachers at their school to participate in this capstone study. As such, they did not have teammates with whom to plan their lessons. Ms. Beck sought the assistance of her ITRT. She taught two lessons that contained references to CT and CT vocabulary. Ms. Jones stated that she wished her teammates had participated so that they could have shared lesson ideas and would have liked to reach out to her Instructional Coaches for support in planning, but did not have time to do so (Jones, first interview).

Ms. Horton and Ms. Sanchez did not reach out to school or district support for assistance because they believed that they already had all of the knowledge and resources that they needed. Nonetheless, Ms. Horton received instructional support from our second interview, almost as an intervention, when she learned that CT was not what she thought it was. With time to process that new information, she explored the content herself. Then she served as an instructional support for her partner teacher, sharing her newfound ideas. While originally dismissive of the challenge to her own perceptions, Ms. Sanchez took time to reconsider and revisit the SOLs for clarification. That conversation between the teaching partners most likely caused a change in Ms. Sanchez's second lesson, but did not result in a lesson that contained instances of CT integration, although it may have been a precursor.

Teachers need instructional support to integrate CT into their instructional lessons. That support can come from their own teammates as they work together to

integrate new required curriculum, or it can come from other school or district-level support staff who have an understanding of CT and the ways to teach CT.

Summary of Implications

The problem of practice of this capstone study focuses on the ways that we can help teachers to implement new standards for computational thinking and infuse the instruction into their lessons. In addressing the research questions of this capstone study, I found that teachers lacked an understanding of and preparedness to teach CT. Therefore, the ways to help teachers implement the new CT standards is to help them develop an understanding of CT and help them to implement high quality CT instruction.

Limitations

Several limitations affect the findings, recommendations, and usefulness of this capstone study. First, my presence during the instructional lessons may have altered the implementation of the lessons. I suspect that the teachers at RFES taught lessons that they thought would relate to my field of work. Additionally, in the interviews that I conducted, my presence may have further caused participants to give the answers they thought were "correct." Second, I did not have participation that was representative of the three schools or RCSD as a whole, so the conclusions and recommendations may not apply to other teachers in RCSD. Third, the findings and recommendations are possibly transferable to other similar research settings, but they are not generalizable to all teachers in RCSD or other public schools in Virginia. Fourth, and similar to the previous limitation, this capstone study focuses only on third grade teachers in RCSD and no other elementary school grades that will also be expected to integrate CT into their instruction. Lastly, because I based my findings and recommendations on my interpretations of events, it is

possible that others who implement this capstone study would arrive at different conclusions and recommendations.

Recommendations

In this section, I present specific recommendations for action on the part of RCSD and the VDOE, as well as the challenges that may impede the implementation of the recommendations. These recommendations focus on the support that these organizations can provide to teachers as they attempt to integrate new standards into their instruction. While the focus of this capstone study is on CT, recommendations apply to the Computer Science Standards of Learning in general, as CT engagement should take place as a part of the integration of these SOLs into instruction. One recommendation is that the VDOE provide curriculum maps for the CS SOLs, similar to those for English, mathematics, or science. The VDOE website indicates that those resources are forthcoming (VDOE, 2019), so I have excluded that recommendation below. A summary of the recommendations is found in Table 16.

Table 16

Summary of Recommendations

Recommendations	
Recommendation 1	VDOE should adopt a clear operational definition of CT for teachers, include a "Computational Thinking and Coding" section on grade-level standards documents, and include related CS standards on content area standards documents and curriculum blueprints.
Recommendation 2	RCSD should make use of available professional development opportunities sponsored by the VDOE on computational thinking in order to develop a strong understanding of computational thinking.
Recommendation 3	RCSD should create a Computer Science SOL Leadership group to drive integration of CT and CS standards into instruction across the district. This group would provide professional development opportunities for teachers to develop a strong understanding of computational thinking and RCSD-specific instructional resources for teachers to implement into high quality computational thinking instruction.

Recommendation One: VDOE should adopt a clear operational definition of CT for teachers, include a "Computational Thinking and Coding" section on grade-level standards documents, and include related CS standards on content area standards documents and curriculum blueprints.

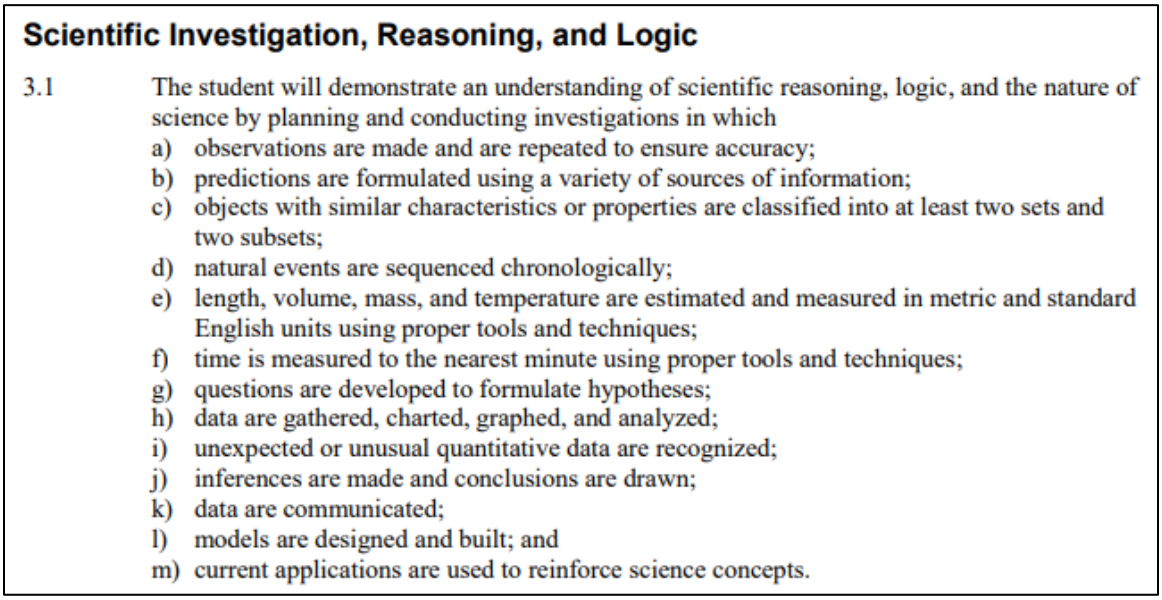
The standards documents as they currently exist do not reduce confusion for teachers about what it is they are expected to teach. First, they should revise the "All Computer Science" document that contains all of the Computer Science Standards of Learning to provide a clear definition of computational thinking. Currently, one section of the document is intended to answer the question of "What is Computational Thinking," but it is the section below on "Computer Science Practice for Students" that offers a definition of CT (see Figure 13). Revising these two sections to make the VDOE's adopted definition of CT clearer should be the first step in remedying the definitional

confusion among teachers looking to integrate CT into their classroom instruction. I propose the VDOE adopt the following as its operational definition of CT for teachers: An approach to solving complex problems logically through abstraction, generalization, decomposition, algorithms, or debugging. These concepts can be used to formulate problems in a way that can be solved by a computer.

Once the VDOE makes this definition clear in the "All Computer Science" document, the definition should be included within each grade-level or stand-alone course standards document as well, as suggested by Ms. Sanchez (second interview). Section headers of the grade-level documents should also change to include specific language used in the revised Code of Virginia related to the Standards of Learning. The code was modified by the Virginia General Assembly to add "computer science and computational thinking, including computer coding" (2016). For example, the header "Algorithms and Programming" should say "Computational Thinking and Coding." These revisions could make the Standards of Learning the only resource a teacher needs to access to find the definition of CT adopted by the VDOE, and teachers wouldn't be forced to search the Internet only to find myriad different definitions of CT.

In addition to revising the Computer Science SOL documents, the VDOE should make a few modifications to the standards documents for core content areas such as English, Mathematics, History & Social Studies, or Science. I do not propose a change to the actual standards themselves at this time, but simply an addition to the text in the document. Currently, the Computer Science SOL documents include suggestions of content area standards in which to integrate specific CS SOLs. For example, CS 3.5 states, "the student will compare and contrast a group of items based on attributes or

actions classified into at least two sets and two subsets. [Related SOL: Science 3.1c]" (VDOE, 2017). I recommend that the third grade Science SOL document add "Related SOL, Computer Science 3.5" to the end of Standard 3.1c. This way, teachers who are planning for their content area lessons might be more likely to add standards from the CS SOLs. Later, as content area SOLs are revised, I recommend including CT as a skill within each domain, similar to the way that all grade-level Science SOLs begin with Standard 1 – Scientific Investigation, Reasoning, and Logic (Figure 15).



Scientific Investigation, Reasoning, and Logic

3.1 The student will demonstrate an understanding of scientific reasoning, logic, and the nature of science by planning and conducting investigations in which

a) observations are made and are repeated to ensure accuracy;
b) predictions are formulated using a variety of sources of information;
c) objects with similar characteristics or properties are classified into at least two sets and two subsets;
d) natural events are sequenced chronologically;
e) length, volume, mass, and temperature are estimated and measured in metric and standard English units using proper tools and techniques;
f) time is measured to the nearest minute using proper tools and techniques;
g) questions are developed to formulate hypotheses;
h) data are gathered, charted, graphed, and analyzed;
i) unexpected or unusual quantitative data are recognized;
j) inferences are made and conclusions are drawn;
k) data are communicated;
l) models are designed and built; and
m) current applications are used to reinforce science concepts.

Figure 15: Standard 1 – Scientific Investigation, Reasoning, and Logic (VDOE, 2010)

The first recommendation focused on the VDOE and the formatting of the Standards of Learning for Computer Science documents, as well as documents related to core subject areas. This recommendation relates to the findings that only three out of eight observed lessons referenced CT and that teachers did not understand CT or have an accurate or consistent definition of it. As mentioned previously, curriculum framework with examples of how the elements of CT and CS can be integrated into disciplinary content are necessary, but also promised by the VDOE in the summer of 2019.

Recommendation Two: RCSD should make use of available professional development opportunities sponsored by the VDOE on computational thinking in order to develop a strong understanding of computational thinking.

Desimone (2009) suggests that the five core features of high quality professional development are content focus, active learning, coherence, duration, and collective participation. The second recommendation takes these factors into consideration as described below.

Currently the VDOE encourages teachers to attend professional development training sessions offered by Code Virginia to learn about content knowledge and instructional strategies for teaching CS and CS SOLs. In the spring of 2018, invitations to participate in a number of summer professional development session were emailed to teachers and school administrators across the state, including in RCSD (personal communication, April 1, 2018). RCSD redistributed one such invitation to attend professional development sessions from Code Virginia through its employee newsletter (personal communication, June 7, 2018), although at the time, teachers had to opt-in to receive updates from the blog. I attended a week-long Code Virginia summer professional development sessions in 2018. Code Virginia continues to offer professional development sessions, and the VDOE is hosting a CSforVA Summit in July 2019 as well.

In preparation for the required implementation of the CS SOLs in the 2019-2020 school year, RCSD should send a targeted group of teachers and administrators to either the Code Virginia sessions or CSforVA Summit. RCSD has content advisory teams led by a central office administrator. The leader of each content advisory group should select a teacher from each of the 15 elementary schools in RCSD, with between two and three teachers per grade level, so that each school has representation and each grade level has

several teachers in attendance. This team of teachers and administrators, the CS SOL Leadership group, forms the basis of my third recommendation, described in the next section.

Beyond the summer of 2019, RCSD should continue to send teachers and administrators to professional development sessions on CT and CS integration in order for those educators to develop a strong understanding of CT. I have limited my suggestions to those sponsored by the VDOE because so far those sessions have been offered free of charge, across a variety of dates and throughout the state, providing more opportunities for teachers to attend throughout the year.

This recommendation addresses Desimone's (2009) proposal of high quality professional development. The professional development sessions by Code Virginia and the VDOE specifically target the content in the new CS SOLs (content focus). When I attended the week-long session by Code Virginia, we engaged in active learning. We worked in groups to design lessons that we practiced teaching on other participants, we lead discussions on our own experiences with learning and teaching CS. We participated in mini-lessons on coding from Scratch and Code.org, and conducted a resource showcase with technology tools available in our own school districts. I introduced the group to the sorting network activity from CSUnplugged.org (see Figure 6). The Code Virginia sessions run for a full week, and include follow-up work online and additional face-to-face sessions throughout the subsequent school year (duration). Collective participation is addressed by sending a full team of RCSD teachers and administrators to these available professional development sessions. Desimone's (2009) recommendation for coherence must be addressed by the VDOE first, to ensure that these professional

development opportunities can target clear objectives, and RCSD teachers will know that they are adhering to the requirements by the state. Without further information, there is no way to know if the professional development offered by Code Virginia or the VDOE will be high quality, but it is possible that it would demonstrate the key features suggested by Desimone (2009).

Recommendation Three: RCSD should create a Computer Science SOL Leadership group to drive integration of CT and CS standards into instruction across the district. This group would provide professional development opportunities for teachers to develop a strong understanding of computational thinking and RCSD-specific instructional resources for teachers to implement into high quality computational thinking instruction.

In forming the CS SOL leadership group described in the second recommendation, RCSD can adopt a model where their own teachers serve as the experts on the new CS SOLs and share that expertise with their colleagues. RCSD has a number of structures in place presently to spread CT and CS pedagogy throughout the district. What follows is a suggested plan for professional development on the new CS SOLs for RCSD teachers.

Phase One: Team Development. The first phase of this professional development plan was described as the second recommendation, that RCSD must assemble a team of teachers and administrators to attend existing state-offered professional development sessions.

Phase Two: District-Wide Session Planning. Central office administrators will use the information that they learned during summer professional development to prepare for an expository session on the new SOLS during the RCSD pre-service teacher week. Teachers return to school for one week before students arrive, and one of those days is dedicated to compulsory, district-wide professional development planned by central

office staff. Administrative members of the CS SOL leadership group should offer one session throughout the day to introduce elementary teachers to CT and their specific CS SOLs and provide teachers with expectations and requirements throughout the year related to the new CS SOLs. Teachers in RCSD likely share misconceptions about CT and CS that participants in this capstone study demonstrated—this is an opportunity to address those misconceptions and to help teachers understand why they should teach CT and CS. Teacher members of the CS SOL leadership group will work alongside the administrators to facilitate the sessions and be introduced district-wide as resources for other teachers in their grade-levels and schools.

Phase Three: Curriculum Development. Grade-level teams from different schools on the CS SOL Leadership group will work together to develop curriculum and lesson plans that integrate CT and the CS SOLs into their content area instruction. Teachers should focus on developing lesson plans for all content areas like mathematics, science, and English, identifying ways that CT and the new CS SOLs can be woven into the content area instruction. They teachers should make use of the resources and instructional strategies that they learned about during the summer, as well as the forthcoming CS SOL curriculum framework documents expected to be released by the VDOE in the summer of 2019. Because these teachers attended professional development that focused on content, engaged them in active learning, was coherent and of a sufficient duration, and built upon collective participation, they should have the necessary understanding of CT and the CS SOLs to do this work. As these teachers develop lessons to teach in their own classrooms, they should share the lessons with their grade-level CS SOL Leadership team so that they can be implemented and critiqued, as well as to build a

repository of lessons across multiple content area domains that integrate CT and the CS standards. Teaching these lessons will give teachers the experience with CT and the CS SOLs, and the necessary opportunity to reflect before sharing their resources with other RCSD teachers (Buchholz et al., 2013). The CS leadership group should reconvene at the end of the first year to review lessons learned and review vertical alignment. This will ensure that teachers and students build on experiences longitudinally as students' skills and knowledge related to CT and CS grow.

Phase Four: Internal Professional Development. RCSD has a number of opportunities for teachers to access internal professional development throughout the year. Three of those opportunities allow district employees to teach courses to their colleagues in exchange for a stipend. The first opportunity is a district-wide professional development day after the first nine-weeks grading period. The second is through after-school courses taught throughout the school year. The third opportunity is during the summer, when teachers must take six hours of internal professional development courses, spread out over two weeks in July, or during one additional professional development day during the pre-service teacher week before students return to school. Teacher members of the CS SOL Leadership group should use these sessions to share the curriculum that they have developed over the school year, as well as any valuable insight that they have gained from their experience integrating CT and the CS SOLs into their content area instruction.

During these PD sessions, attendees must engage in active-learning of CT and CS content knowledge, as well as see examples of the ways CT and CS SOLs can be integrated into content area lessons in RCSD schools. Session facilitators could focus the

instruction they provide on one content area of integration, one element of CT, or one stand of the CS standards, such as Computing Systems, Cybersecurity, or Data Analysis.

For example, a session might focus on ways to integrate the five elements of CT (Angeli et al., 2016) into third grade mathematics instruction. Table 1 provides examples of abstraction and generalization in the third grade math SOLs (VDOE, 2016). Students can create algorithms to describe the process of comparing fractions with unlike denominators to address Standard 3.2b. To practice debugging, teachers could give students a multistep word problem and an incorrect solution to the problem that shows the steps taken to arrive at the wrong solution. The teacher can instruct the students on the ways to identify the errors made, correct those errors, and solve the problem correctly. This addresses Standard 3.3b. Students can engage in decomposition through Standard 3.3b, as well, by correctly solving multistep problems similar to those described above. When students identify the process to solve these multistep problems and complete them in order, they have broken the problem down into smaller parts that are easier to solve. Table 17 provides a summary of these examples and those from Table 1 that address the third grade Standards of Learning for mathematics.

The second and third recommendations address findings of this capstone study related to the instructional methods used to teach CT and access to resources and instructional support when planning for CT integration. By developing a team of CS SOL leaders who have received professional development, RCSD can employ those teachers with a strong understanding of ways to teach CT ad CS to create resources for teachers and provide instructional support and high quality professional development opportunities to their colleagues throughout the year.

Table 17

Examples of Computational Thinking integrated into Grade Three Mathematics Standards of Learning

Element	Example	Standard
Abstraction	Identifying the characteristics of five different rectangles that make them all rectangles but ignoring the ways that the rectangles are different.	Measurement and Geometry – 3.12b
Generalization	Recognizing patterns of "Red-Blue-Red-Red" and "Circle-Square-Circle-Circle" both as ABAA patterns	Patterns, Functions, and Algebra – 3.16
Decomposition	Identifying the parts of a multistep problem that should be added or subtracted first in a multistep problem before moving onto the next parts of the problem	Computation and Estimation – 3.3b
Algorithms	Create a list of steps to follow in order to compare fractions with unlike denominators	Number and Number Sense – 3.2c
Debugging	Identifying and correcting errors made in solving multistep practical problems	Computation and Estimation – 3.3b

Challenges

The implementation of these recommendations presents challenges both for the VDOE and RCSD, particularly time and money. Specifically, the timeliness of these recommendations related to when the new Standards of Learning go into effect is problematic. The VDOE expects teachers to implement the new SOLs in the 2019-2020 school year. In my experience as an educator, we began preparing to integrate new content standards as soon as the VDOE released those standards, one to two years before they were required, in order to practice and iterate on ways to teach the content before it

was required. With the pending implementation months away, there may not be enough time to enact these recommendations.

Time is particularly a factor for the VDOE. The Computer Science SOLS were just adopted in November 2017. It took eight months for a steering committee to develop those standards that were adopted by the Virginia Board of Education (Staples, 2017). The process required to modify even those CS standards based on the recommendations above might take just as long, or longer. Revising the standards for content areas such as English or Science would likely require a similar committee and lengthy timeline as well.

Time would be less of a factor for RCSD. Teachers learn about professional development opportunities year-round. Additionally, because the sessions offered by Code Virginia and the VDOE are in the summer and free of charge, there may be fewer barriers to teacher and administrative participation in those sessions.

Funding for the CS SOL Leadership group may prove challenging for RCSD. Annual budget proposals for the upcoming fiscal years are due months in advance, so there might not be funding in the RCSD professional development budget to stipend these potential teacher-leaders to create curriculum and lead in-service sessions on ways to integrate CT and the CS standards into instruction.

Summary of Recommendations

I based these recommendations for the VDOE and RCSD on the findings of this capstone study in response to the research questions. The recommendations encourage the VDOE to clarify the meaning of computational thinking and its place in the Computer Science Standards of Learning. The recommendations also suggest RCSD to leverage available professional development sessions to enhance the understanding of content and

ways to teach that content for a team of teachers and administrators that will create instructional resources and provide instructional support to their colleagues as they work to integrate CT and the CS SOLs into instructional lessons. Chapter Six includes the action communication in which I present both the findings and recommendations to the VDOE and RCSD.

Action Communication I

To: Timothy Ellis
Computer Science and Virtual Learning Specialist
Virginia Department of Education

From: Dr. Albert H. Jacoby, III, Ed.D.
University of Virginia
4370 Saddle Court
Earlysville, VA 22936

Date: July 1, 2019

Dear Sir:

I am writing to report findings and recommendations based on an 11-week single-case study of four third grade teachers in the Rockview County School District (RCSD) as they attempted to integrate computational thinking (CT) into their content area instruction. During the study, I observed two lessons by each teacher, interviewed the teachers after each lesson, and collected documents for review.

As you know, the Virginia General Assembly modified the Code of Virginia related to the Standards of Learning to include "computer science and computational thinking, including computer coding" in 2016. The result of that modification was new Standards of Learning in Computer Science, adopted in November 2017 and scheduled for enactment during the 2019-20 school year. As a part of my research, I approached teachers, told them of the new SOLs, and asked to observe their practices to explore the ways that they planned for and integrated CT into their curriculum.

The findings and recommendations of this study can help make informed decisions about the Standards of Learning and the ways that teachers should prepare for their implementation in the fall of 2019. This case study is specific to third grade teachers at four elementary schools in RCSD, and are not generalizable to all teachers in RCSD or the Commonwealth of Virginia. The recommendations and findings are starting points for further exploration of best practice.

The findings of the study are as follows:

1. Only three of the eight lessons taught by teachers in RCSD as a part of this capstone study contained elements of CT. Each of the three touched upon one

or more element of CT identified by Angeli et al. (2016) to varying degrees, and collectively touched upon all five elements.

2. Teachers whose lessons contained elements of CT used direct, didactic instruction in order to integrate CT into their instruction. Teachers focused on helping students understand CT terms and vocabulary by relating the concepts to students' everyday lives.

3. Teachers in RCSD did not have a common, shared understanding of the meaning of CT, as defined by the elements identified by Angeli et al. (2016) or otherwise. Teachers suffered from definitional confusion related to CT, and struggled to make sense of their own interpretations of CT, even when provided with concrete definitions and relevant examples.

4. The lessons that touched upon elements of CT were taught by two teachers who used CT resources. Resources used include lesson plans, articles, graphics, and instructional videos. Two teachers who did not touch upon elements of CT did not use CT resources.

5. One teacher who taught lessons that touched upon elements of CT used district support personnel. Those personnel accessed and modified CT resources and co-planed the lessons with the teacher.

Based on the findings above, I recommend the following actions for the Virginia Department of Education (VDOE) in order to better support teachers' implementation of CT through the new Computer Science Standards of Learning.

Recommendation One: VDOE should adopt a clear, operational definition of CT for teachers, include a "Computational Thinking and Coding" section on grade-level standards documents, and include related CS standards on content area standards documents and curriculum blueprints.

The standards documents as they currently exist do not reduce confusion for teachers about what it is they are expected to teach. First, they should revise the "All Computer Science" document that contains all of the Computer Science Standards of Learning to make the definition of computational thinking more clear. Currently, one section of the document is intended to answer the question of "what is computational thinking," but it is the section below on "computer science practices for students" that offers a definition of CT (see Figure 13). Revising these two sections to make the VDOE's adopted definition of CT clearer could help to remedy some of the definitional confusion among teachers looking to integrate CT into their classroom instruction. I recommend that the VDOE adopt the following as an operational definition of CT for teachers: An approach to solving complex problems logically, through abstraction, generalization, decomposition, algorithms, or debugging. These concepts can be used to formulate problems in a way that can be solved by a computer.

Once the VDOE makes this definition clear in the "All Computer Science" document, the definition should be included within each grade-level or stand-alone course standards document as well. Section headers of the grade-level documents should also change to include specific language used in the revised Code of Virginia related to the Standards of Learning. The code was modified by the Virginia General Assembly to add "computer science and computational thinking, including computer coding" (2016). For example, the header "Algorithms and Programming" should say "Computational Thinking and Coding." These revisions could make the Standards of Learning the only resource a teacher needs to access to learn the definition of CT, and teachers wouldn't have to search the Internet only to find myriad different definitions of CT.

In addition to revising the Computer Science SOL documents, the VDOE should make a few modifications to the standards documents for core content areas such as English, Mathematics, History & Social Studies, or Science. I do not propose a change to the actual standards themselves, but simply an addition to the text in the document. Currently, the Computer Science SOL documents include suggestions of content area standards in which to integrate specific CS SOLs. For example, CS 3.5 states, "the student will compare and contrast a group of items based on attributes or actions classified into at least two sets and two subsets. [Related SOL: Science 3.1c]" (VDOE, 2017). I recommend that the third grade Science SOL document add "Related SOL, Computer Science 3.5" to the end of Standard 3.1c. This way, teachers who are planning for their content area lessons might be more likely to add standards from the CS SOLs.

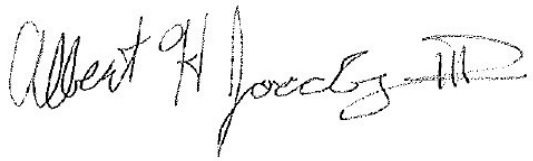
There are additional recommendations based on the findings that are specific to RCSD. Those recommendations relate to professional development and curriculum development specific to CT and the Computer Science SOLs.

Several limitations affect the findings, recommendations, and usefulness of this capstone study. First, my presence during the instructional lessons may have altered the implementation of the lessons. I suspect that some teachers taught lessons that they thought would relate to my field of work. Additionally, in the interviews that I conducted, my presence may have further caused participants to give the answers they thought were "correct." Second, I did not have participation that was representative RCSD or the State of Virginia, so the conclusions and recommendations may not apply to other teachers. Third, the findings and recommendations are possibly transferable to other similar research settings, but they are not generalizable to all teachers in RCSD or other public schools in Virginia. Fourth, and similar to the previous limitation, this capstone study focuses only on third grade teachers in RCSD and no other elementary school grades that will also be expected to integrate CT into their instruction. Lastly, because I based my findings and recommendations on my interpretations of events, it is possible that others who implement this capstone study would arrive at different conclusions and recommendations.

I hope that these findings and recommendations will be useful to you and the Virginia Department of Education. I would be happy to provide an abbreviated list of references regarding computational thinking, as well as a list of available computational

thinking resources. Please do not hesitate to contact me if you have questions or concerns. You can reach me at my e-mail address, ahj2yc@virginia.edu.

Sincerely,



Albert H. Jacoby, III.

Action Communication II

To: Director of Elementary Education
Rockview County School District

From: Dr. Albert H. Jacoby, III, Ed.D.
University of Virginia
4370 Saddle Court
Earlysville, VA 22936

Date: July 1, 2019

Dear Madam:

I am writing to report findings and recommendations based on an 11-week single-case study of four third grade teachers in the Rockview County School District (RCSD) as they attempted to integrate computational thinking (CT) into their content area instruction. During the study, I observed two lessons by each teacher, interviewed the teachers after each lesson, and collected documents for review.

As you know, the Virginia General Assembly modified the Code of Virginia related to the Standards of Learning to include "computer science and computational thinking, including computer coding" in 2016. The result of that modification was new Standards of Learning in Computer Science, adopted in November 2017 and scheduled for enactment during the 2019-20 school year. As a part of my research, I approached teachers, told them of the new SOLs, and asked to observe their practices to explore the ways that they planned for and integrated CT into their curriculum.

The findings and recommendations of this study can help make informed decisions about the Standards of Learning and the ways that teachers should prepare for their implementation in the fall of 2019. This case study is specific to third grade teachers at four elementary schools in RCSD, and are not generalizable to all teachers in RCSD. The recommendations and findings are starting points for further exploration of best practice.

The findings of the study are as follows:

6. Only three of the eight lessons taught by teachers in RCSD as a part of this capstone study contained elements of CT. Each of the three touched upon one or more element of CT identified by Angeli et al. (2016) to varying degrees, and collectively touched upon all five elements.

7. Teachers whose lessons contained elements of CT used direct, didactic instruction in order to integrate CT into their instruction. Teachers focused on helping students understand CT terms and vocabulary by relating the concepts to students' everyday lives.

8. Teachers in RCSD did not have a common, shared understanding of the meaning of CT, as defined by the elements identified by Angeli et al. (2016) or otherwise. Teachers suffered from definitional confusion related to CT, and struggled to make sense of their own interpretations of CT, even when provided with concrete definitions and relevant examples.

9. The lessons that touched upon elements of CT were taught by two teachers who used CT resources. Resources used include lesson plans, articles, graphics, and instructional videos. Two teachers who did not touch upon elements of CT did not use CT resources.

10. One teacher who taught lessons that touched upon elements of CT used district support personnel. Those personnel accessed and modified CT resources and co-planed the lessons with the teacher.

Based on the findings above, I recommend the following actions for the Virginia Department of Education (VDOE) and Rockview County School District in order to better support teachers' implementation of CT through the new Computer Science Standards of Learning.

Recommendation One*:* VDOE should adopt a clear, operational definition of CT for teachers, include a "Computational Thinking and Coding" section on grade-level standards documents, and include related CS standards on content area standards documents and curriculum blueprints.

The standards documents as they currently exist do not reduce confusion for teachers about what it is they are expected to teach. First, they should revise the "All Computer Science" document that contains all of the Computer Science Standards of Learning to make the definition of computational thinking more clear. Currently, one section of the document is intended to answer the question of "what is computational thinking," but it is the section below on "computer science practices for students" that offers a definition of CT (see Figure 13). Revising these two sections to make the VDOE's adopted definition of CT clearer could help to remedy some of the definitional confusion among teachers looking to integrate CT into their classroom instruction. I recommend that the VDOE adopt the following as an operational definition of CT for teachers: An approach to solving complex problems logically, through abstraction,

generalization, decomposition, algorithms, or debugging. These concepts can be used to formulate problems in a way that can be solved by a computer.

Once the VDOE makes this definition clear in the "All Computer Science" document, the definition should be included within each grade-level or stand-alone course standards document as well. Section headers of the grade-level documents should also change to include specific language used in the revised Code of Virginia related to the Standards of Learning. The code was modified by the Virginia General Assembly to add "computer science and computational thinking, including computer coding" (2016). For example, the header "Algorithms and Programming" should say "Computational Thinking and Coding." These revisions could make the Standards of Learning the only resource a teacher needs to access to learn the definition of CT, and teachers wouldn't have to search the Internet only to find myriad different definitions of CT.

In addition to revising the Computer Science SOL documents, the VDOE should make a few modifications to the standards documents for core content areas such as English, Mathematics, History & Social Studies, or Science. I do not propose a change to the actual standards themselves, but simply an addition to the text in the document. Currently, the Computer Science SOL documents include suggestions of content area standards in which to integrate specific CS SOLs. For example, CS 3.5 states, "the student will compare and contrast a group of items based on attributes or actions classified into at least two sets and two subsets. [Related SOL: Science 3.1c]" (VDOE, 2017). I recommend that the third grade Science SOL document add "Related SOL, Computer Science 3.5" to the end of Standard 3.1c. This way, teachers who are planning for their content area lessons might be more likely to add standards from the CS SOLs.

Recommendation Two: RCSD should make use of available professional development opportunities sponsored by the VDOE on computational thinking in order to develop a strong understanding of computational thinking.

Currently the VDOE encourages teachers to attend training sessions offered from Code Virginia to learn about content knowledge and instructional strategies for teaching CS and CS SOLs. Code Virginia continues to offer professional development sessions, including during the summer of 2019, and the VDOE is hosting a CSforVA Summit in July 2019 as well.

In preparation for the required implementation of the CS SOLs in the 2019-2020 school year, RCSD should send a targeted group of teachers and administrators to either the Code Virginia sessions or CSforVA Summit, or both. The leader of each content advisory team should select a teacher from each elementary schools in RCSD, with between two and three teachers per grade level, so that each school has representation and each grade level has several teachers in attendance. This team of teachers and administrators, the CS SOL Leadership group, forms the basis of my third recommendation, described in the next section.

Beyond the summer of 2019, RCSD should continue to send teachers and administrators to professional development sessions on CT and CS integration. I have limited my suggestions to those sponsored by the VDOE because so far those sessions have been offered free of charge, across a variety of dates and throughout the state, providing more opportunities for teachers to attend throughout the year.

Recommendation Three*:* RCSD should create a Computer Science SOL Leadership group to drive integration of CT and CS standards into instruction across the district. This group would provide professional development opportunities for teachers to develop a strong understanding of computational thinking and RCSD-specific instructional resources for teachers to implement into high quality computational thinking instruction.

In forming the CS SOL leadership group described in the second recommendation, RCSD can adopt a model where their own teachers serve as the experts on the new CS SOLs and share that expertise with their colleagues. This will allow RCSD to take advantage of its numerous professional development structures in place presently to spread CT and CS pedagogy throughout the district. What follows is a suggested plan for introducing professional development on integrating the CS SOLs into instruction for RCSD teachers.

*Phase One: Team Development***.** The first phase of this professional development plan was described as the second recommendation. RCSD must assemble a team of teachers and administrators to attend existing state-offered professional development sessions.

*Phase Two: District-Wide Session Planning***.** Content advisory team leaders will use the information that they learned during summer professional development to prepare for an expository session on the new SOLS during the RCSD pre-service teacher week. Content advisory team leaders should offer one session throughout the day to introduce elementary teachers to CT and their specific CS SOLs and provide teachers with expectations and requirements throughout the year related to the new CS SOLs. Teachers in RCSD likely share misconceptions about CT and CS—this is an opportunity to address those misconceptions and to help teachers understand why they should teach CT and CS. Teacher members of the CS SOL leadership group will facilitate the sessions alongside the team leaders and be introduced district-wide as resources for other teachers in their buildings or grade-levels.

*Phase Three: Curriculum Development***.** Grade-level teams within the CS SOL Leadership group will work together in order to develop curriculum and lesson plans that integrate CT and the CS SOLs into their content area instruction. Teams should focus on developing lesson plans for all content areas, identifying ways that CT and the new CS SOLs can be woven into the content area instruction. They should make use of the resources and instructional strategies that they learned about during the summer, as well as CS SOL Curriculum Framework documents expected to be released by the VDOE in the summer of 2019 (VDOE, 2019). As these teachers develop lessons to teach in their own classrooms, they should share the lessons with their grade-level CS SOL Leadership

group so that they can be implemented and critiqued, as well as to build a repository of lessons across multiple content area domains that integrate CT and the CS standards. The CS leadership group should reconvene at the end of the first year to review lessons learned and review vertical alignment. This will ensure that teachers and students build on experiences longitudinally as students' skills and knowledge related to CT and CS grow.

Phase Four: Internal Professional Development. As you know, RCSD has a number of opportunities for teachers to access internal professional development throughout the year. Three of those opportunities allow district employees to teach courses to their colleagues in exchange for a stipend. The first opportunity is the district-wide professional development day after the first nine-weeks grading period. The second is through after-school courses taught throughout the school year. The third opportunity over two weeks in July or during the pre-service teacher week. Teacher members of the CS SOL Leadership group should use these sessions to share the curriculum that they have developed over the school year, as well as any valuable insight that they have gained while integrating CT and the CS SOLs into their content area instruction.

Attendees must engage in active-learning of CT and CS content knowledge, and see examples of the ways that CT and CS SOLs have been integrated into instruction in RCSD schools. Session facilitators should focus their instruction on one content area for integration, one element of CT across multiple content areas, or one stand of the CS Standards, such as Computing Systems, Cyber Security, or Algorithms and Programming, across multiple content areas.

Several limitations affect the findings, recommendations, and usefulness of this capstone study. First, my presence during the instructional lessons may have altered the implementation of the lessons. I suspect that some teachers taught lessons that they thought would relate to my field of work. Additionally, in the interviews that I conducted, my presence may have further caused participants to give the answers they thought were "correct." Second, I did not have participation that was representative RCSD, so the conclusions and recommendations may not apply to other teachers. Third, the findings and recommendations are possibly transferable to other similar research settings, but they are not generalizable to all teachers in RCSD. Fourth, and similar to the previous limitation, this capstone study focuses only on third grade teachers in RCSD and no other elementary school grades that will also be expected to integrate CT into their instruction. Lastly, because I based my findings and recommendations on my interpretations of events, it is possible that others who implement this capstone study would arrive at different conclusions and recommendations.

I hope that these findings and recommendations will be useful to Rockview County School District. I would be happy to provide an abbreviated list of references regarding computational thinking, as well as a list of available computational thinking resources. Please do not hesitate to contact me if you have questions or concerns. You can reach me at my e-mail address, ahj2yc@virginia.edu.

Sincerely,

Dr. Albert H. Jacoby, III, Ed.D.

REFERENCES

Aho, A. V. (2011). What is computation? *Ubiquity,* 1–8. https://doi.org/doi: 10.1007/bf00413693

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), 47–57.

Armoni, M., & Gal-ezer, J. (2014). Early Computing Education— WHY? WHAT? WHEN? WHO? *ACM Inroads*, *5*(4), 54–59. https://doi.org/10.1145/2684721.2684734

Assembly, V. G. Code of Virginia - Standards of Quality (2016). United States. Retrieved from http://law.lis.virginia.gov/vacode/22.1-253.13:1

Bell, T., Witten, I. H., Fellows, M., Adams, R., & McKenzie, J. (2015). *CS Unplugged: An enrichment and extension programme for primary-aged children*. Retrieved from http://csunplugged.org/wp-content/uploads/2015/03/CSUnplugged_OS_2015_v3.1.pdf

Bers, M. U. (2010). The TangibleK Robotics program: Applied computational thinking for young children. *Early Childhood Research & Practice, 12*(2), 1–20.

Bers, M. U., Flannery, L., Kazakoff, E. R., & Sullivan, A. (2014). Computational thinking and tinkering: Exploration of an early childhood robotics curriculum. *Computers and Education*, *72*, 145–157. https://doi.org/10.1016/j.compedu.2013.10.020

BrainPOP. (2019). Computational Thinking - BrainPOP Jr. Retrieved November 29, 2018, from

https://jr.brainpop.com/artsandtechnology/technology/computationalthinking/

Brannan, K., Chung, M., Martin, W., Cervates, F., Tally, B., & Resnick, M. (2019). Computational Thinking with Scratch. Retrieved June 20, 2019, from http://scratched.gse.harvard.edu/ct/index.html

Brennan, K., Balch, C., & Chung, M. (2011). *Creative computing*. *Harvard Graduate School of Education*. Cambridge, Massachusetts. https://doi.org/10.1016/S0022-3913(12)00047-9

Brennan, K., & Resnick, M. (2012). New frameworks for studying and assessing the development of computational thinking. In *Annual American Educational Research Association meeting, Vancouver, BC, Canada* (pp. 1–25). Retrieved from http://web.media.mit.edu/~kbrennan/files/Brennan_Resnick_AERA2012_CT.pdf

Buchholz, M., Seli, M., & Schulte, C. (2013). PCK and reflection in computer science teacher education. *Proceedings of the 8th Workshop in Primary and Secondary Computing Education*, 8–16. https://doi.org/10.1145/2532748.2532752

Buitrago Flórez, F., Casallas, R., Hernández, M., Reyes, A., Restrepo, S., & Danies, G. (2017). Changing a generation's way of thinking: Teaching computational thinking through programming. *Review of Educational Research*, *87*(4), 834–860. https://doi.org/10.3102/0034654317710096

Catrow, V. (2016, May 16). Governor adds computer science to Virginia SOLs. *RVANews*, pp. 1–2. Retrieved from https://rvanews.com/etc/governor-adds-computer-science-to-virginia-sols/137501

Cetin, I., & Dubinsky, E. (2017). Reflective abstraction in computational thinking. *Journal of Mathematical Behavior*, *47*(June), 70–80.

https://doi.org/10.1016/j.jmathb.2017.06.004

Code.org. (2019a). About Us | Code.org. Retrieved June 20, 2019, from https://code.org/about

Code.org. (2019b). Code.org - Learn Computer Science. Retrieved June 20, 2019, from http://studio.code.org/courses

Code.org. (2019c). Real-ife Algorithms | Paper Airplanes. Retrieved June 20, 2019, from https://code.org/curriculum/course2/2/Teacher

Coe, I. R., & Ferworn, A. (2016). The life and contributions of Countess Ada Lovelace. *IEEE Technology and Society Magazine*, *35*(4), 46–49.

Cohen, D., & Crabtree, B. (2006). Qualitative Research Guidelines Project. Retrieved December 23, 2017, from http://www.qualres.org/HomeLinc-3684.html

Council, N. R. (2010). *Report of a Workshop on The Scope and Nature of Computational Thinking*. (T. N. A. Press, Ed.). Washington, D.C. https://doi.org/10.17226/12840

Council, N. R. (2011). *Report of a Workshop on the Pedagogical Aspects of Computational Thinking*. (T. N. A. Press, Ed.). Washington, D.C.: National Research Council. https://doi.org/10.17226/13170

Csizmadia, A., Curzon, P., Humphreys, S., Ng, T., Selby, C., & Woollard, J. (2015). Computational thinking: A guide for teachers.

Denning, P. J. (2009). Beyond computational thinking. *Communications of the ACM*, *52*(6), 28. https://doi.org/10.1145/1516046.1516054

Denning, P. J. (2017). Remaining trouble spots with computational thinking. *Communications of the ACM*, *60*(6), 33–39. https://doi.org/10.1145/2998438

Denning, P. J., Comer, D. E., Gries, D., Mulder, M. C., Tucker, A., Turner, A. J., &

Young, P. R. (1989). Computing as a Discipline. *Communications of the ACM*, *32*(I), 9–23. https://doi.org/10.1007/978-1-4614-3987-5

Desimone, L. M. (2009). Improving impact studies of teachers' professional development: Toward better conceptualizations and measures. *Educational Researcher*, *38*(3), 181–199. https://doi.org/10.3102/0013189X08331140

Education, C. S. (2017). Strong Passwords. San Francisco: Common Sense Education. Retrieved from https://www.commonsense.org/education/lesson/strong-passwords-3-5

Education, U. S. D. of, Statistics, N. C. for E., & (HEGIS), H. E. G. I. S. (2012). Degrees in computer and information sciences conferred by degree-granting institutions, by level of degree and sex of student: 1970-71 through 2010-11. *Degrees and Other Formal Awards Conferred Surveys*. Retrieved from https://nces.ed.gov/programs/digest/d12/tables/dt12_349.asp

Education, V. D. of. (2013). Computer technology standards of learning for Virginia's public schools grades 9-12, (February), 10.

Education, V. D. of. (2016). Grade Three Mathematics Standards of Learning for Virginia Public Schools - September 2016, (January), 4–6.

Education, V. D. of. (2017). *Computer science Standards of Learning for Virginia public Schools*. Richmond, VA. Retrieved from http://www.doe.virginia.gov/testing/sol/standards_docs/computer-science/2017/stds-compsci-all.pdf

Education, V. D. of. (2019). Frequently Asked Questions. Richmond: Virginia Department of Education. Retrieved from

http://www.doe.virginia.gov/instruction/computer-science/index.shtml

Fessakis, G., Gouli, E., & Mavroudi, E. (2013). Problem solving by 5-6 years old kindergarten children in a computer programming environment: A case study. *Computers and Education*, *63*, 87–97. https://doi.org/10.1016/j.compedu.2012.11.016

Fluck, A., Webb, M., Cox, M., Angeli, C., Malyn-smith, J., & Zagami, J. (2016). Arguing for computer science in the school curriculum. *Educational Technology & Society*, *19*(3), 38–46.

Furber, S. (2012). *Shut down or restart? The way forward for computing in UK schools*. *The Royal Society*. London, UK. Retrieved from https://royalsociety.org/~/media/education/computing-in-schools/2012-01-12-computing-in-schools.pdf

Go, S., & Dorn, B. (2016). Thanks for sharing: CS pedagogical content knowledge sharing in online environments. *Proceedings of the 11th Workshop in Primary and Secondary Computing Education- WiPSCE '16*, 27–36. https://doi.org/10.1145/2978249.2978253

Google. (n.d.-a). Computational Thinking for Education - - Unit 2 - Exploring Algorithms. Retrieved June 20, 2019, from https://computationalthinkingcourse.withgoogle.com/unit?lesson=6&unit=2

Google. (n.d.-b). Computational Thinking for Education - Course. Retrieved June 20, 2019, from https://computationalthinkingcourse.withgoogle.com

Google. (n.d.-c). Google for Education: Computational Thinking. Retrieved June 20, 2019, from http://www.google.com/edu/computational-thinking

Google. (2013). Google Ngram Viewer. Retrieved June 20, 2019, from https://books.google.com/ngrams

Grgurina, N., Barendsen, E., Zwaneveld, B., van Veen, K., & Stoker, I. (2014). Computational thinking skills in Dutch secondary education: Exploring pedagogical content knowledge. *Proceedings of the 14th Koli Calling International Conference on Computing Education Research (Koli Calling '14)*, 173–174. https://doi.org/10.1145/2674683.2674704

Grossman, P. L. (1989). A study in contrast: sources of pedagogical content knowledge for secondary english. *Journal of Teacher Education*, *40*(5), 24–31.

Grover, S., & Pea, R. (2013). Computational thinking in K-12: A review of the state of the field. *Educational Researcher*, *42*(1), 38–43. https://doi.org/10.3102/0013189X12463051

Hattie, J. (2009). *Visible Learning: A Synthesis of Over 800 Meta-Analyses Relating to Achievement*. Abingdon: Routledge.

Heiten, L. (2016). Va. Gov. Signs K-12 Computer Science Bill, Making the Subject a Requirement for All. Retrieved July 23, 2016, from http://blogs.edweek.org/edweek/curriculum/2016/05/virginia_governor_signs_k-12_computer_science_bill.html

Ilic, U., Haseski, H. I., & Tugtekin, U. (2018). Publication trends over 10 Years of computational thinking research. *Contemporary Educational Technology*, *9*(2), 131–153. https://doi.org/10.30935/cet.414798

IMACS. (2012). Computational Thinking for Kids. Retrieved January 30, 2019, from https://www.eimacs.com/blog/2012/04/computational-thinking-for-kids/

International Society for Technology in Education (ISTE). (2016). *ISTE Standards for Students*. Eugene, OR: International Society for Technology in Education.

International Society for Technology in Education (ISTE), & Computer Science Teachers Association (CSTA). (2011a). *Computational Thinking Leadership Toolkit*. Computer Science Teachers Association (CSTA) and the International Society for Technology in Education (ISTE). Retrieved from http://www.iste.org/docs/ct-documents/ct-leadershipt-toolkit.pdf?sfvrsn=4%0Ahttp://www.iste.org/learn/computational-thinking

International Society for Technology in Education (ISTE), & Computer Science Teachers Association (CSTA). (2011b). *Computational Thinking Teacher Resources* (2nd ed.). Computer Science Teachers Association (CSTA) and the International Society for Technology in Education (ISTE). Retrieved from http://search.ebscohost.com/login.aspx?direct=true&db=bth&AN=9609037773&site=ehost-live

Israel, M., Pearson, J. N., Tapia, T., Wherfel, Q. M., & Reese, G. (2015). Supporting all learners in school-wide computational thinking: A cross-case qualitative analysis. *Computers and Education*, *82*, 263–279. https://doi.org/10.1016/j.compedu.2014.11.022

K12cs.org. (2016). *K–12 Computer Science Framework.* Retrieved from https://k12cs.org/wp-content/uploads/2016/09/K–12-Computer-Science-Framework.pdf

Kjellstrom, W. (2017). *Enhancing elementary preservice teachers' technological, pedagogical, and content knowledge through an alternatively designed course*.

University of Virginia.

Knuth, D. E. (1972). George Forsythe and the development of computer science. *Communications of the ACM*, *15*(8), 721–726. https://doi.org/10.1145/361532.361538

Kumar, D. (2014a). Digital playgrounds for early computing education. *ACM Inroads*, *5*(1), 20–21. https://doi.org/10.1145/2568195.2568200

Kumar, D. (2014b). Welcome. *ACM Inroads*, *5*(4), 52–53. https://doi.org/10.1145/2684721.2684733

Lester, F. K. (2013). Thoughts about research on mathematical problem-solving instruction. *The Mathematics Enthusiast*, *10*(10), 1551–34401. https://doi.org/ISSN 1551-3440

Little, J. W. (1987). Teachers as colleagues. In V. Richardson-Koehler (Ed.), *Educators' handbook: A research perspective* (pp. 491–518). New York: Longman.

Lye, S. Y., & Koh, J. H. L. (2014). Review on teaching and learning of computational thinking through programming: What is next for K-12? *Computers in Human Behavior*, *41*, 51–61. https://doi.org/10.1016/j.chb.2014.09.012

MakerEd. (2019). What is Maker Education? Retrieved March 19, 2019, from https://makered.org/about/what-is-maker-education/

Maloney, J., Peppler, K., Kafai, Y. B., Resnick, M., & Rusk, N. (2008). Programming by choice: urban youth learning programming with scratch. In *SIGCSE '08 Proceedings of the 39th SIGCSE technical symposium on Computer science education* (pp. 367–371). https://doi.org/10.1145/1352135.1352260

Margolis, J., Estrella, R., Goode, J., Holme, J. J., & Nao, K. (2008). *Stuck in the shallow*

end. Cambridge, Massachusetts: The MIT Press.

Marshall, C., & Rossman, G. B. (2011). *Designing qualitative Research* (Fifth). Los Angeles: SAGE Publications, Inc.

Martinez, M. E. (2006). What is metacognition? *Source: The Phi Delta Kappan*, *87*(9), 696–699. Retrieved from http://www.jstor.org/stable/20442131

Martinez, S. L., & Stager, G. (2013). *Invent to learn: Making, tinkering, and engineering in the classroom*. Torrance, CA: Constructing Modern Knowledge Press.

Miles, M. B., & Huberman, A. M. (1994). *Qualitative data analysis: An expanded sourcebook* (Second). Thousand Oaks, CA: SAGE Publications, Inc.

National Science Teachers Association. (2013). *Practice 5: Using Mathematics and Computational Thinking*. Retrieved from http://nstahosted.org/pdfs/ngss/MatrixOfScienceAndEngineeringPractices.pdf

NCTM. (2010). Why is teaching with problem solving important to student learning? *National Council of Teachers of Mathematics*, *13*(12), 1–6. https://doi.org/10.1016/S2213-8587(14)70016-6

Nelson, M., Sahami, M., & Wilson, C. (2016). A new framework to define K-12 computer science education. *Communications of the ACM*, *59*(4), 12.

Obama, B. (2016). Weekly address: Giving every student an opportunity to learn through Computer Science for All. United States: YouTube. Retrieved from https://obamawhitehouse.archives.gov/photos-and-video/video/2016/01/29/weekly-address-giving-every-student-opportunity-learn-through-comp

Ouahbi, I., Kaddari, F., Darhmaoui, H., Elachqar, A., & Lahmine, S. (2015). Learning basic programming concepts by creating games with Scratch programming

environment. *Procedia - Social and Behavioral Sciences*, *191*, 1479–1482. https://doi.org/10.1016/j.sbspro.2015.04.224

Owen, S., Repenning, A., Webb, D. C., Brand, C., Gluck, F., Grover, R., … Song, M. (2014). Beyond Minecraft: Facilitating computational thinking through modeling and programming in 3D. *Computer Graphics and Applications*, *34*(3), 68–71.

Papert, S. (1980). *Mindstorms: Children, computers, and powerful ideas*. New York: Basic Books.

Partovi, H. (n.d.). *Testimony of Hadi Partovi, Co-founder and CEO, Code.org*. Washington, D.C. Retrieved from https://www.appropriations.senate.gov/imo/media/doc/hearings/Code.org - OWT.pdf

Pólya, G. (1957). *How to Solve it: A New Aspect of Mathematical Method* (2d ed.). Garden City, N.Y.: Doubleday. https://doi.org/10.2307/3609122

Prottsman, K. (2014). Computer Science for the Elementary Classroom. *ACM Inroads*, *5*(4), 60–63. https://doi.org/10.1145/2684721.2684735

Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., … Kafai, Y. (2009). Scratch: Programming for all. *Communications of the ACM*, *52*(11), 60–67. https://doi.org/10.1145/1592761.1592779

Sáez López, J. M., González, M. R., & Cano, E. V. (2016). Visual programming languages integrated across the curriculum in elementary school: A two year case study using “Scratch” in five schools. *Computers & Education*, *97*, 129–141. https://doi.org/10.1016/j.compedu.2016.03.003

Saunders, M. R. (2017). *First review of the proposed 2017 Computer Science Standards*

of Learning. Richmond, Virginia. Retrieved from http://www.doe.virginia.gov/boe/meetings/2017/07-jul/agenda-items/item-f.pdf

Schoenfeld, A. H. (1985). *Mathematical Problem Solving*. Orlando, FL: Academic Press. Retrieved from http://math-dept.talif.sch.ir/pdf/manaba/%5BAlan_Schoenfeld%5D_Mathematical_Problem_Solving.pdf

Seehorn, D., Carey, S., Fuschetto, B., Lee, I., Moix, D., O'Grady-Cuniff, D., … Verno, A. (2011). *CSTA K-12 Computer Science Standards*. *CSTA Standards Task Force*. Retrieved from http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:CSTA+K+–+12+Computer+Science+Standards#1

Selby, C. C. (2014). *How can the teaching of programming be used to enhance computational thinking skills?* University of Southampton. https://doi.org/10.1109/fie.2016.7757408

Slavin, T. (2014). What is computational thinking? Retrieved January 30, 2019, from https://www.kidscodecs.com/computational-thinking-projects/

Staples, S. R. (2017). Schedule for development of computer science standards of learning. *Superintendent's Memo*, *036*(17), 3–4.

Statistics, B. of L. (2019). Computer and Information Technology Occupations. Retrieved June 19, 2019, from https://www.bls.gov/ooh/computer-and-information-technology/home.htm

Szász, D. (2011). John von Neumann, the Mathematician. *Mathematical Intelligencer*, *33*(2), 42–51. https://doi.org/10.1007/s00283-011-9223-6

Tolboom, J., Krüger, J., & Grgurina, N. (2014). *Informatica in de bovenbouw havo / vwo. SLO - Nationaal expertisecentrum leerplanontwikkeling*. Enschede. Retrieved from http://www.slo.nl/downloads/2014/informatica-in-de-bovenbouw-havo-vwo.pdf/

Turing, A. M. (1950). Computing machinery and intelligence. *Mind*, *59*(236), 433–460. https://doi.org/http://dx.doi.org/10.1007/978-1-4020-6710-5_3

Unplugged, C. (2019). The Book - Computer Science Unplugged. Retrieved June 20, 2019, from http://csunplugged.org/books

van Del, J., Verloop, N., & de Vos, W. (1998). Developing science teachers' pedagogical content knowledge. *Journal of Research in Science Teaching*, *35*(6), 673–695. https://doi.org/10.1002/(SICI)1098-2736(199808)35:6<673::AID-TEA5>3.0.CO;2-J

van Driel, J. H., & Berry, A. (2012). Teacher professional development focusing on pedagogical content knowledge. *Educational Researcher*, *41*(1), 26–28. https://doi.org/10.3102/0013189X11431010

Virginia Department of Education. (2010). Grade Three Science Standards of Learning for Virginia Public Schools - 2010, 1–8. Retrieved from http://www.doe.virginia.gov/testing/sol/standards_docs/science/2010/courses/stds_Earth_science.pdf

Voogt, J., Fisser, P., Good, J., Mishra, P., & Yadav, A. (2015). Computational thinking in compulsory education: Towards an agenda for research and practice. *Education and Information Technologies*, *20*(4), 716–728. https://doi.org/10.1007/s10639-015-9412-6

Voskoglou Michael Gr.; Buckley, S., Voskoglou, M. M. G., & Buckley, S. (2012). Problem solving and computational thinking in a learning environment. *ArXiv*

Preprint ArXiv:1212.0750, *36*(May 2014), 28–46. Retrieved from http://arxiv.org/abs/1212.0750

Wing, J. M. (2006). Computational Thinking. *Communications of the Association for Computing Machinery (ACM)*, *49*(3), 33–35. https://doi.org/https://www.cs.cmu.edu/~15110-s13/Wing06-ct.pdf

Wing, J. M. (2010). *Computational Thinking: What and Why?*

Woodcock, J. (2016). *Coding Games in Scratch*. New York, NY: DK Children.

Yadav, A., Good, J., Voogt, J., & Fisser, P. (2017). Computational thinking as an emerging competence domain. In M. Mulder (Ed.), *Competence-based Vocational and Professional Education* (pp. 1051–1067). Springer International Publishing. https://doi.org/10.1007/978-3-319-41713-4

Yadav, A., Mayfield, C., Zhou, N., Hambrusch, S., & Korb, J. T. (2014). Computational Thinking in Elementary and Secondary Teacher Education. *ACM Transactions on Computing Education*, *14*(1), 1–16. https://doi.org/10.1145/2576872

Yadav, A., Stephenson, C., & Hong, H. (2017). Computational thinking for teacher education. *Communications of the ACM*, *60*(4), 55–62. https://doi.org/10.1145/2994591

Yin, R. K. (2014). *Case Study Research*. Thousand Oaks, CA: SAGE Publications, Inc.

APPENDIX A

ECT Pencil Code Program: Map Visualization

At a glance…

Core subject(s)	History Social Science; Computer Science
Subject area(s)	US History; Programming Fundamentals
Suggested age	8 to 18 years old

Overview

Use this program to show a simple way to illustrate statistics geographically by drawing bubbles on a map. Have students analyze or fill in or change parts of the program. This program could be used to further your understanding of how you could use Pencil Code in the classroom, as a demonstration or discussion with your students, or as a way to introduce various **CT concepts**, such as pattern recognition or abstraction, to your students by inviting them to extend the existing functionality of the program.

Pencil Code Program

Copy/Paste the following program into a 'Blank Editor' on the Pencil Code website (new.pencilcode.net)

```
# Copyright 2015 Google Inc. All Rights Reserved.

# Licensed under the Apache License, Version 2.0 (the "License");
# you may not use this file except in compliance with the License.
# You may obtain a copy of the License at

# http://www.apache.org/licenses/LICENSE-2.0

# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.

Somalia = new Turtle
Somalia.wear 'http://goo.gl/4E8BcV'
Somalia.css 'z-index', -1
moveto 110, 80
dot yellow, 30
label "Eyl"
moveto -25, -105
dot skyblue, 70
label "Merca"
ht()
```

Sample Output

Additional Information and Resources

Computational Thinking Concepts*

Concept	Definition
Abstraction	Identifying and extracting relevant information to define main idea(s)
Pattern Recognition	Observing patterns, trends, and regularities in data

** Explore the Computational Thinking Concepts Guide for a list of the CT concepts noted on ECT, including tips for implementing each concept in your classroom*

Additional Resource Links

- Visit http://pencilcode.net/ to explore the Pencil Code development environment
- See Pencil Code: A Programming Primer for more than 100 example programs written in CoffeeScript

Administrative Details

Contact info For more info about Exploring Computational Thinking (ECT), visit the ECT website (g.co/exploringCT)

Credits	Developed by the Exploring Computational Thinking team at Google and reviewed by K-12 educators from around the world.
Last updated on	06/09/2015
Copyright info	

APPENDIX B

ECT Lesson Plan: Making Music with Algorithms

Lesson plan at a glance...

Core subject(s)	Music; Computer Science
Subject area(s)	Music; Algorithms and Complexity
Suggested age	11 to 18 years old
Prerequisites	*None*
Time	**Preparation:** 5 to 10 minutes **Instruction:** 60 minutes
Standards	**CS:** CSTA 2-7, 9; 3A-3, 12

In this lesson plan…

Lesson Overview

In almost every culture, people of all ages are making and listening to music, whether it is by banging hands rhythmically against a table or composing an elaborate symphony with all types of instruments. Music is a product of each of our cultures, but are there patterns and general principles that apply to most, if not all, music? In this lesson, students will **recognize patterns** in music and modify an **algorithm** by **analyzing data** to improve the quality of the music generated by a Pencil Code program.

Materials and Equipment

- For the teacher:
  a) *Required:* Presentation set-up
     i) Internet-connected computer
        (1) Chrome browser (https://www.google.com/chrome/browser/desktop) recommended
     ii) External speakers for audio playback
  b) *Required:* Access to YouTube (http://www.youtube.com)
  c) *Recommended:* Whiteboard and dry-erase markers or equivalent
- For the student:
  a) *Required:* Internet-connected computers (one (1) computer per student recommended)
  a) *Required:* Pencil Code Music Maker program
  b) *Required:* Journal
     i) Google Docs (http://docs.google.com) or a wiki
     OR
     ii) Markers/Whiteboard or Paper and Pen/Pencil
  c) Required: Headphones

Preparation Tasks

	Confirm that computers are on, logged-in, and connected to the Internet with working audio (speakers, headphones)	5 to 10 minutes

The Lesson

Warm-up Activity: Factors involved in making music	10 minutes
Activity 1: Patterns in musical scales	10 minutes
Activity 2: Adjusting an algorithm to generate music	30 minutes
Wrap-up Activity: Reflecting on making music with algorithms	10 minutes

Warm-up Activity: Factors involved in making music (10 minutes)
Activity Overview: In this activity, students will explore some of the characteristics of music and examples of what humans do to create music.

Activity:
Journaling: Students respond to the following prompts in their journal or word processor:
Prompt 1: What are some reasons why humans create music?

Prompt 2: If you were going to make music with only your hands and feet what might you do to make the music interesting?

- After a couple of minutes, ask students to share their answers.
- Write their responses on the board.
- Use check marks to indicate and count duplicate responses.

Activity 1: Patterns in musical scales (10 minutes)
Activity Overview: In this activity, students will recognize patterns in scales from around the world and how changing the notes can affect its listeners.

Notes to the Teacher:
The number line is used to illustrate music scales in a way that does not require prior knowledge of music notation. To play scales, you can use Wikipedia's Music Scale (https://en.wikipedia.org/wiki/List_of_musical_scales_and_modes) or search on YouTube for an example (https://www.youtube.com/watch?v=QDWKzG5oaog) [C major scale].

You could supplement the lesson with these videos:

a) Universality of the Pentatonic scale (https://www.youtube.com/watch?v=ne6tB2KiZuk)
b) Ccommon chords found in many popular songs (https://www.youtube.com/watch?v=pCrD9N_3Jkw)

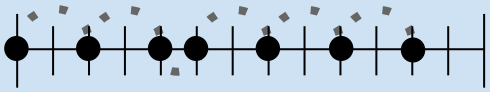
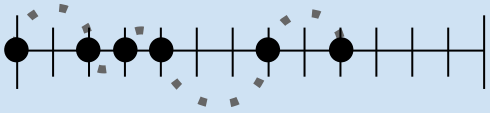
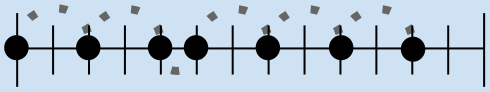
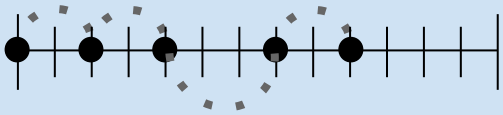
Activity:

- Play a few scales found around the world (see the Teacher notes above for guidance). Have students write down on a rating scale of 1-10 how much they liked the scale as well as a few adjectives to describe how the scale made them feel. Do not tell the students what the names of the scales are when you are playing them, refer to them as scale #1, scale #2, etc.

Scale	Rating of how much you enjoyed it (1-10)	Adjectives you would use to describe how the scale makes you feel
#1		
#2		
#3		
#4		
#5		

- Have students share the adjectives they used to describe each scale.
- Now have students look at a diagram of the scale and look for a correlation between the pattern in the scales and how much they enjoyed that scale.

Example diagram of specific scales:

C major scale	
Blues	
Japanese	
Pentatonic	

Activity 2: Adjusting an algorithm to generate music (30 minutes)
Activity Overview: In this activity, students will modify an existing **algorithm** by adjusting parameters to generate music that sounds pleasant to them. Students can compare the adjustments they made to the program with each other to begin to understand the general principles of music theory through **data analysis**.

Notes to the Teacher:
Students do not need to have any prior understanding of music theory or notation in order to work on this program. The background on each topic is provided in the activity below for you to use as needed.

It is not necessary to modify the algorithm used to play the notes. Once the activity is complete, time permitting, students may modify the algorithm to add more keys or any additional features that they choose.

Activity:
Walk students through the following:

- Open the Pencil Code Music Maker program (https://example.pencilcode.net/edit/music-maker#blocks=0).
- This program creates and displays two pianos, randomly generating and playing notes based on a pentatonic scale. Students can modify parts of the algorithm to explore and figure out which settings they prefer.
- Start by modifying line 24 to change the scale from `notes = pentatonic` to `cmajor`, `hungarian`, or another listed on lines 6 through 12. This changes what notes are available to play.
  a) For those familiar with musical scales it will be helpful to know that in Pencil Code, a sharp is indicated with a ^, or caret, preceding the letter (e.g. ^D is D sharp, or D♯) and a flat note has an _, or underscore preceding the letter (e.g. _A is A flat or A ♭).
- Once you have settled on a set of notes to use, modify the type of beats used in the composition.
  a) A beat establishes the rhythm for a song. When you are listening to a song you may find yourself clapping along or tapping your foot and these are beats. If you were clapping 4 times every second while singing it would sound very different than if you were only clapping 1 time per second. Musicians have a common language so they can write down music and understand what each other is saying. Some examples of these notes are whole, half, quarter, eighth are the most common.
  b) The rhythm of the main tune (or *melody*) of a song is often expressed by a singer or an instrument, such as a drum or bass guitar. In the default code, the melody is set to play quarter notes which play for 1 beat. Modify the number of beats in the melody by including different types of beats. For example `melodyBeats = [quarter, whole]` would result in music with some notes held only for one beat (quarter note) and others held for the whole measure or

four beats (whole note). The options can be found on lines 16 through 20.

c) In the default code, the rhythm is set to play whole notes which resonate for 4 beats. Modify the number of beats in the rhythm by including different types of beats. For example `rhythmBeats = [half, whole]`.

- The tempo sets the rate at which beats are played. The default in the code is 130 beats per minute, you can increase or decrease this to change the energy of the song.
- These three attributes, notes or scale, rhythm, and tempo are core components of music. Try modifying one of these while keeping the other two constant. You may discover how dramatically these attributes can affect the song. For example:
  a) the scale can make a song sound like it is happy, sad, or even frightening, it can be thought of as words used in a story
  b) the beats are like the sentences in a story and are used to convey an emotion. The rhythm can make you want to dance, sit in somber reflection, or simply tap your foot.
  c) tempo can change the energy of a song temporarily to cause excitement or tension or follow a certain genre or style.

Q1: In the table below indicate what values you would use to make a song convey certain emotions. Feel free to test this out in Pencil Code by modifying the values for `tempo`, `melodyBeats`, and `bassBeats`.

Emotion	`tempo`	`melodyBeats`	`bassBeats`
Happy			
Sad			
Calm			
_____ (fill in your own)			

- For every beat of the song in this program, a note is randomly selected from the notes available (you previously set this in line 24, the notes variable) and all notes by default have an equal probability. A composer of music does not randomly select notes, they use experience and feeling to determine which note to play next. You can modify the probability of each note by changing the values in `biasForNotes`. You can change the number to be any value between 0 and 1 and ideally the values would all add up to 1.
- Change the values in `biasForNotes` to see which pattern works best, you may prefer to give some values a high probability and others have little or zero chance of being selected. Record in the

table below the values for `biasForNotes` you preferred.

Preferences	`biasForNotes` values
#1	[, , , , , ,]
#2	[, , , , , ,]
#3	[, , , , , ,]

Q2: What values for `biasForNotes` produced the most enjoyable music for you?

- Now that you have had some time to adjust the algorithm, setup the program to play music in the way that is preferable to you. Share your insights with other students. This could be specific discoveries such as, "I prefered these notes and rhythm," or general patterns like, "The music sounded best to me when the probability of the notes was not the same."
- If you would like to continue to modify the algorithm to improve your composition, the table below has some additional parameters that can be modified. Adjust one parameter at a time so you can understand the effect each one has on the music.

Line	Name	Description	Suggestions
23	Pianos	The number of pianos generated. Currently there are two, a piano to play the melody and a piano to play the bass.	Add a harmony piano to play accompany the melody `pianos = [melody, harmony, bass]`
25	Measures	The number of measures determines the length of a song. The algorithm defines a measure as 4 beats long so 5 measures will create a song 20 beats long.	Increase/decrease the number of measures `measures = 10`
29	Melody Octaves	The range of notes available for the melody. An octave is a complete set of notes and the distance between a note and one that sounds the same, but just higher or lower. For example there is a note A at a frequency of 440Hz and one above it at 880Hz.	Add/remove octaves using Pencil Code notation of commas and single quotation marks. `melodyOctaves = ["", "'", "''"]`

30	Bass Octaves	The range of notes available for the bass (see Melody Octaves for more). Bass notes sound 'deeper' than the higher notes in the Melody Octaves.	Add/remove octaves using Pencil Code notation of commas and single quotation marks. `bassOctaves = [",,", ","]`

Teaching Tips:

- If students are struggling to come up with ways to modify the code, write suggestions on the board or pause halfway in to share what others have tried and ask what would happen if we did _____.
- Hint, a popular value for `biasForNotes` is `[0.33, 0 , 0 , 0.33 , 0.33 , 0 , 0]`.

For students who would like to continue to modify the algorithm they could:

- Add additional scales, modify the time signature (https://en.wikipedia.org/wiki/Time_signature), and other refinements to the existing algorithm.
- Modify the algorithm used to play the notes found on lines 33 - 89. Modifying the algorithm could be used, for example, to give each measure a new set of rules (perhaps to introduce a solo [https://en.wikipedia.org/wiki/Solo_(music)] halfway into the music).

Assessment:

A1: Answers will vary:

- happy - the `tempo` should be high `melodyBeats` could include at least quarter notes and possibly eighth notes, `bassBeats` may include half notes and quarter notes.
- sad - the `tempo` may be low `melodyBeats and bassBeats` could include whole and half notes

A2: Answers will vary.

Wrap-up Activity: Reflecting on making music with algorithms (10 minutes)

Activity Overview: In this activity, students will reflect on any new techniques they learned for creating music as well as ideas for improving the music generating algorithm.

Activity:

Journaling: Students respond to the following prompts in their journal or word processor:

Prompt 1: What additional ideas have you come up with for creating music using only your hands and feet (compare with your list from the beginning of this lesson).

Prompt 2: Even if you don't know how you might implement these ideas, what do you think would improve the music generated by the algorithm in the previous activity?

- After a couple of minutes, ask students to share their answers.
- Write their responses on the board.
- Use check marks to indicate and count duplicate responses.

Teaching Tips:

- Show students examples of videos found on YouTube when you search for [algorithm generated music] or [computer generated music] to see examples of how others have attempted to generate music using an algorithm. Ask students how they think the music was made.

Learning Objectives and Standards

Learning Objectives	**Standards**
LO1: Students will analyze musical scales for patterns and preference.	*Computer Science* CSTA 2-7: Represent data in a variety of ways: text, sounds, pictures, numbers. CSTA 2-9: Interact with content-specific models and simulations to support learning and research.
LO2: Students will be able to create music by modifying an algorithm.	*Computer Science* CSTA 2-9 CSTA 3A-3: Explain how sequence, selection, iteration, and recursion are building blocks of algorithms. CSTA 3A-12: Describe how computation shares features with art and music by translating human intention into an artifact.

Additional Information and Resources

Lesson Vocabulary

Term	**Definition**	**For Additional Information**
Multiple musical terms were defined in activity 2 to assist students in modifying the algorithm.		
Rhythm	A pattern of regular, repeated sounds	https://en.wikipedia.org/wiki/Rhythm
Scale	A set of musical notes ordered by frequency or pitch.	https://en.wikipedia.org/wiki/Scale_(music)

Step or Interval	The distance between two notes in a scale.	https://en.wikipedia.org/wiki/Scale_(music)
Time Signature	The number of beats per measure and the note value which equals 1 beat.	https://en.wikipedia.org/wiki/Time_signature

Computational Thinking Concepts

Concept	Definition
Algorithm Design	Creating an ordered series of instructions for solving similar problems
Data Analysis	Making sense of data by finding patterns or developing insights
Pattern Recognition	Observing patterns and regularities in data

Additional Resource Links

- WolframTones Musical Composition Generator - requires QuickTime (http://tones.wolfram.com/generate/)

Administrative Details

Contact info	For more info about Exploring Computational Thinking (ECT), visit the ECT website (g.co/exploringCT)
Credits	Developed by the Exploring Computational Thinking team at Google and reviewed by K-12 educators from around the world.
Last updated on	07/24/2015
Copyright info	

APPENDIX C

Divide and conquer

- **Duration**: 30 minutes
- **Ages 8 to 10:** Lesson 2

Classroom resources

- Paper
- Payment system such as tokens or marbles
- Pens

Learning outcomes

Students will be able to:

- Describe how the time taken grows with the size of the input, and most importantly how it grows in different ways for two different algorithms. Computational Thinking: Generalising and Patterns
- Describe how to compare number values for equality and inequality (greater than, less than). Mathematics: Numeracy
- Explain how they used decomposition to divide and conquer when doing a binary search. Computational Thinking: Decomposition
- Explain the range of the number of guesses for unsorted lists compared to sorted lists. Mathematics: Statistics
- Identify search algorithms for sorted and unsorted lists (sequential and binary search). Computer Science: Algorithms

Key questions

Imagine 31 numbers have been organised in ascending order in a list by a computer program. Now the program has to find a number in the list, but it can only look at one number at a time. Is it easier to find the number now, than if they were in a random order?

Potential answers could include:

- The information is organised in a way that makes it more efficient to find. By guessing one number and checking it, you can use logical thinking to eliminate the numbers below or above the number guessed because you know the numbers are in order.

- If they are in random order you can't use a strategy to find them quickly.

Lesson starter

We have 31 different numbers, one on each card. You can't see them but this time they are in order from the lowest number to the highest number. The numbers range from 0 to 1000. Can you find number **302**

Teaching observations

You can adapt the range to suit what your students are working on in their mathematics lessons. This lesson focuses on sorted lists and we are using a range of numbers from 0 - 999. This activity works best if the numbers aren't sequential because, for example, if there are 50 cards in the range 1 to 50 and they are sequential and you ask a student to find the number 10 they will probably just look at the 10th card straight away! You can generate different sets of cards with various ranges of numbers here. It's best if the numbers aren't spread evenly so that it's very hard to guess where a particular value might be.

Lesson activities

Set up a line of cards, with the animal facing upwards. Have a payment system ready such as tokens for your classroom, counters, sweets, or marbles.

The game is even better if you have some real stakes - for example: I have 10 marbles each is worth 2 minutes of game time. For every token you use to find the number I'm thinking of, you will lose a marble.

Let's see how many guesses it takes to find the number: **302**

Who would like the first guess? (Choose a student). Which animal should I turn over? Tell us why you chose that guess. (They should be selecting the card that is exactly half way. If they didn't, check with the class if they agree with the choice or could they add to the student's thinking to select the middle card. If they decide not to, that's fine; they will learn the hard way if they use a less efficient approach.)

Turn over the chosen card to show the number under it. If it's the correct one, you can stop, otherwise *remove that card and all other cards that the number can't be* (which will either be all cards to the right or all cards to the left of the chosen one) and take away one token from your pile. Repeat this process until a student chooses the card with your number on it; if they use all ten tokens then the teacher "wins".

How many guesses did it take to find the number?

With each guess, how many cards were eliminated from being a possibility? (Answer: half the cards could be eliminated with each guess if you picked the middle card.)

Did the students win because they guessed within 10 guesses or did you win because it took them longer?

Repeat this game until the students have won 3 times or you have won 3 times.

Teaching observations

The number of guesses required can be anything from 1 (if you are lucky the first time), to 5 (if you have chosen the middle number each time). Of course, they may use more than 5 guesses if they use a poor strategy. Most of the time they will need close to 5 guesses. This also means that students will always have tokens left if they use a good strategy, since the maximum number of guesses to find the number is 5.

Applying what we have just learnt

If any data is organised in order and a binary search is applied, then you can eliminate a lot of data quickly - cutting the number of items in half each time. As a slightly different example, if we were trying to guess a number between 1 and 1,000,000, then asking if the number is over 500,000 would eliminate 500,000 options in one question, the second question eliminates 250,000, the third question 125,000, and so on. So in 3 questions you have eliminated 875,000 numbers. With just 20 questions you can find the one number between 1 and 1,000,000. It's the same searching for objects that are sorted in descending order - each value that is checked halves the number of possible locations. Dividing problems in half makes them very small very quickly.

This general process is called "divide and conquer" - you break the problem into (two) parts, and deal with each part separately, in turn break them into two parts. Very soon you end up with very easy tasks, such as dealing with just one item. It's a great strategy for reducing any big task or challenge to achievable goals!

Lesson reflection

What is the algorithm for a binary search? Here is a possible answer:

- Ask to see the middle card
- Repeat until the correct number is found:
- Is the number greater than the number I want to find?
    - If yes, then keep the cards above that number,
    - Else, keep the cards below that number

Seeing the Computational Thinking connections

Throughout the lessons there are links to computational thinking. Below we've noted some general links that apply to this content.

Teaching computational thinking through CSUnplugged activities supports students to learn how to describe a problem, identify what are the important details they need to solve this problem, and break it down into small, logical steps so that they can then create a process which solves the problem, and then evaluate this process. These skills are transferable to any other curriculum

area, but are particularly relevant to developing digital systems and solving problems using the capabilities of computers.

These Computational Thinking concepts are all connected to each other and support each other, but it's important to note that not all aspects of Computational Thinking happen in every unit or lesson. We've highlighted the important connections for you to observe your students in action. For more background information on what our definition of Computational Thinking is [see our notes about computational thinking].

Algorithmic thinking 

The divide and conquer process of repeatedly checking the centre card and deducing which cards can be eliminated, and which ones could still contain the number you are searching for, can be written as an algorithm. When you ask students to say which card to check each time they are actually articulating an algorithm and instructing you on how to follow it.

By describing this method with the following algorithm a computer or person can follow it without needing to know how it works, they can just follow the instructions and not have to think about how to actually do the task. It's important that algorithms are written like this, because computers can't figure out how to solve problems by themselves! A possible version of the algorithm is written under the lesson reflection.

Examples of what you could look for:

Who are the students who not only can explain the exact process to find the number, but are also the students who don't deviate from that process?

Abstraction 

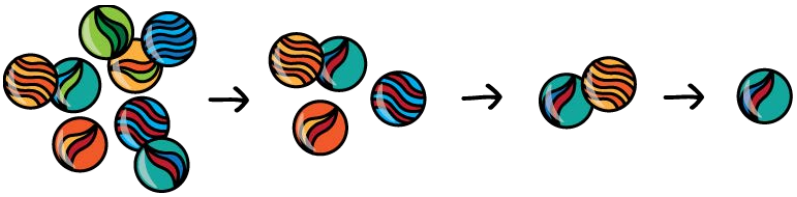
We can use the divide and conquer approach for more problems than just searching through an ordered list. We can use it to search through any set of objects that have identifying features.

We can also use it to help us sort things into order, which will be explored in the Sorting Algorithms unit.

Examples of what you could look for:

If you repeat this exercise but with the numbers underneath different objects, or maybe use different letters of the alphabet or different coloured discs to search for, who are the students that can see that these differences don't actually matter and they are still solving the same problem?

Decomposition 



The Divide and Conquer method is entirely about decomposition. When we use divide and conquer to solve a problem, we are breaking the problem in half repeatedly, which soon decomposes it to a very simple case: a list of one item, which is very easy to search!

Examples of what you could look for:

Who are the students who are able to break the problem down into steps and then explain why each step is important?

Generalising and patterns

The key pattern to recognise in this activity is the process of eliminating half the possible cards by only looking in one, and that this is repeated over and over to accomplish the task.

Like we talked about in the lesson plan, the divide and conquer strategy is a pattern that appears frequently in computer science, and also in real life! It is an efficient and logical way of attacking many different problems where you are searching for something in a group of objects that have different identifying features.

Examples of what you could look for:

Who are the students who quickly identified the pattern?

Evaluation

Students can evaluate how well the divide and conquer method works by looking at how many marbles (or whatever payment you decide to use) they have left at the end of the activity. Older students can further examine the efficiency of this algorithm by calculating the maximum number of checks it would make for a different numbers of cards. You could compare this to the number of checks that a sequential search would need, and how these numbers change as you increase the number of cards.

Examples of what you could look for:

Who are the students who can explain the strengths and potential problems of using a binary search to find data? Can they explain why the maximum number of checks a binary search will make is much smaller than the maximum for sequential search?

Logic

To retain as many marbles (or whatever payment you decide to use) as possible it makes sense to try and eliminate as many cards as possible with each guess. That way you can cut down the number of cards to 1 as fast as possible. Students will generally be able to logically reason and recognise that the best way to do this is to check the centre card each time. If we check that card and compare it to the number we are searching for what does that tell us about all the cards to the left of that card? All the cards to the right of that card? Students can deduce which cards they can now eliminate based on the card they have checked.

Asking students to explain how they came to this conclusion is a great way to exercise their thinking skills, by getting them to articulate the logical steps they followed to come to this conclusion, and why it makes sense that doubling the number of cards only needs 1 more check. Understanding why divide and conquer will only ever require a specific small number of steps at most (for example it will never take more than 5 checks for 19 cards, or 20 checks for 1,000,000 cards) also requires a high level of logical reasoning.

Examples of what you could look for:

Which students instinctively go for the middle square when searching? They are likely logical thinkers who can deduce that since the numbers are sorted then the middle square will tell them the most useful information.

Printables

Searching Cards
Includes helper sheet for teacher.

Teacher guide sheet for binary search activity

Use this sheet to circle the number you are asking your class to look for when you are demonstrating how the binary search works. This allows you to demonstrate the maximum number of searches it would take. When students are playing the number hunt game, they can choose any number. Avoid those numbers that are underlined as they are key binary search positions (avoiding them is a good thing to do for demonstrations, but in practice students, or computers, won't intentionally avoid these).

Sorted numbers

1 to 15

1 - 2 - 3 - 4 - 5 - 6 - 7 - 8 - 9 - 10 - 11 - 12 - 13 - 14 – 15

8	7	6	5
			

4	3	2	1
			

15	14	13
		

12	11	10	9
			

APPENDIX D

Teacher Interview Protocol – First Observation

Introduction

Hello, and thank you for agreeing to meet and speak with me today. I am conducting research on instruction of computational thinking at Rusty Falls Elementary School. The purpose of this interview is to discuss your integration of computational thinking into instruction in the lesson I observed today, both in the lesson planning and the lesson implementation. I will also ask about your experience with computational thinking. I will also gather information about your professional background. Is it OK if I audio record the interview today? I will transcribe the interview and any personally identifying information about you and the research site will be replaced with a pseudonym. Would you like to pick the pseudonym that will be used for you? Do you have any questions for me before we begin?

Observation Summary

First, I am going to ask you questions about the lesson that you taught today.

- Please describe the lesson that I observed today.
    a) What was the subject and content area of the lesson taught?
    b) How long did it take?
    c) What instructional practices or strategies did you use?
    d) What about this lesson specifically addressed computational thinking?
    e) In what ways were the students engaged in computational thinking throughout this lesson?
    f) How were students assessed?
- Why did you choose to teach this lesson for me to observe today?

- What did you notice about students in this lesson?
- What difficulties did students experience when learning computational thinking in this lesson?

Lesson Planning

- How did you plan and prepare for this lesson?
    a) How long did it take to plan for this lesson?
    b) Who did you plan it with?
    c) What instructional materials did you access while planning?
    d) What computational thinking did you want me to see in this lesson the way that you planned it?
    e) How did administration influence you during the planning of this lesson?
    f) How did district support staff influence you during the planning of this lesson?
    g) What would have been helpful during planning of this lesson?
- How did you deviate from your original lesson plan, if at all?
    a) What caused you to deviate from your original lesson plan?
- What will you do differently, if anything, the next time that you plan/prepare for and teach this lesson?

Professional Experience/Background Questions

- Please tell me about your educational background.
    a) Where did you attend college?
    b) What degrees have you earned and are currently pursuing?
    c) How did you earn teaching licensure/certification?
- Please tell me about your professional experience.

a) How long have you been teaching?

b) How long have you been teaching at Rusty Falls Elementary School?

c) What grades and/or classes do you currently teach?

d) How long have you been in your current teaching assignment?

e) What grades and/or classes have you previously taught?

f) How long were you in your previous teaching assignments, if applicable?

g) What other assignments or responsibilities do you have at Rusty Falls Elementary School or in Rockview County School District?

- Please tell me about yourself.

a) What is your age?

b) What is your gender?

Wrap-up

- Do you have any questions?

Thank you for allowing me to observe your instruction and interview you today. If we have not already done so, would you like to schedule your second observation and interview now?

APPENDIX E

Teacher Interview Protocol – Second Observation

Introduction

Hello, and thank you for agreeing to meet and speak with me again today. As a reminder, I am conducting research on instruction of computational thinking at Rusty Falls Elementary School. The purpose of this interview is to discuss your integration of computational thinking into instruction in the lesson I observed today, both in the lesson planning and the lesson implementation. I will also ask about your experience with computational thinking. Is it OK if I audio record the interview today? I will transcribe the interview and any personally identifying information about you and the research site will be replaced with a pseudonym. Do you have any questions for me before we begin?

Observation Summary

First, I am going to ask you questions about the lesson that you taught today.

- Please describe the lesson that I observed today.
    a) What was the subject and content area of the lesson taught?
    b) How long did it take?
    c) What instructional practices or strategies did you use?
    d) What about this lesson specifically addressed computational thinking?
    e) In what ways were the students engaged in computational thinking throughout this lesson?
    f) How were students assessed?
- Why did you choose to teach this lesson for me to observe today?
- What did you notice about students in this lesson?

- What difficulties did students experience when learning computational thinking in this lesson?

Lesson Planning

- How did you plan and prepare for this lesson?
    a) How long did it take to plan for this lesson?
    b) Who did you plan it with?
    c) What instructional materials did you access while planning?
    d) What computational thinking did you want me to see in this lesson the way that you planned it?
    e) How did administration influence you during the planning of this lesson?
    f) How did district support staff influence you during the planning of this lesson?
- How did you deviate from your original lesson plan, if at all?
    a) What caused you to deviate from your original lesson plan?
- What will you do differently, if anything, the next time that you plan/prepare for and teach this lesson?
    a) What grades and/or classes do you currently teach?
    b) How long have you been in your current teaching assignment?
    c) What grades and/or classes have you previously taught?
    d) How long were you in your previous teaching assignments, if applicable?
    e) What other assignments or responsibilities do you have at Rusty Falls Elementary School or in Rockview County School District?

Computational Thinking

- How do you define computational thinking?

- What importance do you assign to the teaching of computational thinking?
    a) Is it important to teach computational thinking?
    b) Why/why not?

Please review this table (see Table 1) with definitions of computational thinking.

- How do you provide students with the opportunities to learn and practice these skills identified as elements of computational thinking?
- In your prior experience, what difficulties do students have when learning these skills identified as elements of computational thinking?
- What do you think students need to know in order for them to learn these skills identified as elements of computational thinking?
- What are your specific ways of assessing students understanding or abilities at these skills identified as elements of computational thinking?
- How do other teachers provide your students with opportunities to learn and practice these skills identified as elements of computational thinking?

Instructional Support related to Computational Thinking

- How does school support staff at Rusty Falls Elementary School support teachers as they teach computational thinking?
- How does the Rockview County School District support staff support teachers as they teach computational thinking?

Besides the ways mentioned before about school and district support, in what other ways are teachers supported as they teach computational thinking?

Wrap-up

- Do you have any questions?

Thank you for participating in this research study, for allowing me to observe your instruction, and interview you today.

APPENDIX F

Computer Science Standards of Learning – Grade Three

The standards for third grade place an emphasis on decomposing larger problems and utilizing the iterative design process to develop a plan to construct and execute programs. Students in third grade are introduced to using computing systems to model attributes and behaviors associated with a concept. The accurate use of terminology as well as the responsible use of technology will continue to be built upon. The foundational understanding of computing and the use of technology will be an integral component of successful acquisition of skills across content areas.

Algorithms and Programming

3.1 The student will construct sets of step-by-step instructions (algorithms), both independently and collaboratively
- using sequencing;
- using loops (a wide variety of patterns such as repeating patterns or growing patterns); and [Related SOL: Math 3.16]
- using events.

3.2 The student will construct programs to accomplish tasks as a means of creative expression using a block or text based programming language, both independently and collaboratively
- using sequencing;
- using loops (a wide variety of patterns such as repeating patterns or growing patterns); and
- identifying events.

3.3 The student will analyze, correct, and improve (debug) an algorithm that includes sequencing, events, and loops. [Related SOL areas – Math: Problem Solving, English: Editing]

3.4 The student will create a plan as part of the iterative design process, independently and/or collaboratively using strategies such as pair programming (e.g., storyboard, flowchart, pseudo-code, story map. [Related SOL: English 3.8c]

3.5 The student will compare and contrast a group of items based on attributes or actions classified into at least two sets and two subsets. [Related SOL: Science 3.1c]

3.6 The student will break down (decompose) a larger problem into smaller sub-problems, independently or collaboratively. [Related SOL: Math 3.3b]

3.7 The student will give credit to sources when borrowing or changing ideas (e.g., using information and pictures created by others, using music created by others, remixing programming projects). [Related SOL: English 3.10e]

Computing Systems

3.8 The student will model how a computing system works including input and output. [Related SOL: Math 3.16]

3.9 The student will identify, using accurate terminology, simple hardware and software problems that may occur during use, and apply strategies for solving problems (e.g., rebooting the device, checking for power, checking network availability, closing and reopening an app).

Cybersecurity

3.10 The student will identify problems that relate to inappropriate use of computing devices and networks.

3.11 The student will create examples of strong passwords, explain why strong passwords should be used, and demonstrate proper use and protection of personal passwords.

Data and Analysis

3.12 The student will answer questions by using a computer to observe data in order for the student to draw conclusions and make predictions. [Related SOL: Math 3.15, HSS 3.1d]

3.13 The student will create an artifact using computing systems to model the attributes and behaviors associated with a concept (e.g., day and night, animal life cycles, plant life cycles). [Related SOL areas – Math: Models, Science: Moon Phases]

Impacts of Computing

3.14 The student will identify computing technologies that have changed the world and express how those technologies influence, and are influenced by, cultural practices.

3.15 The student will identify the positive and negative impacts of the pervasiveness of computers and computing in daily life (e.g., downloading videos and audio files, electronic appliances, wireless Internet, mobile computing devices, GPS systems, wearable computing).

3.16 The student will identify social and ethical issues that relate to computing devices and networks. [Related SOL: C/T: 6-8.3, HSS 3.11]

Networking and the Internet

3.17 The students will discuss in partners and as a class that information can be transmitted using computing devices via a network (e.g., email, blogging, video messaging).

APPENDIX G

Script for Consent Agreement Meeting

Good morning, thank you for allowing me to attend your grade level meeting. My name is Bert Jacoby, and I am conducting research on Computational Thinking as a part of my doctoral capstone project. I am here today to introduce my project to you and hope to gain your consent to conduct research in your classrooms.

In 2016, the Virginia legislators modified the Code of Virginia regarding the Standards of Learning to include Computer Science and Computational Thinking, including coding. Last fall, the Virginia department of education approved new state Standards of Learning in Computer Science to address the change in the law. These standards go into effect in the beginning of the next school year, 2019-2020.

My research is on how teachers prepare for and implement these new standards, particularly on Computational Thinking. I would like to observe each of your classrooms two times, and on that same day conduct a follow-up interview with you about the lesson that I observed and how you prepared for it. I would ask that each lesson observed be of a different content area—reading, writing, math, science, social studies, etc.

During my observations, I would be taking descriptive field notes about your instruction, your interaction with your students, and the instructional activities that your students engage in. No personally identifiable information would be collected, as well as no data about your students, other than observations of the work that they do in class. I will provide you with a notification letter to send home with your students. My observations, as well as the interviews, would be audio recorded.

You would provide your consent by signing a consent document, but if at any time you decide to change your mind and withdraw your consent, you may do so and you and your observations and interviews would no longer be a part of my study.
You will not be compensated for this study in any way, and there are no foreseeable risks or benefits to participating in the study. I do hope, however, that the results of my research prove to be useful to teachers and school districts who seek to integrate Computational Thinking into their instruction.

If you are willing to participate, I will provide you with a consent form to sign. For those of you who consent to participate, I will email you in one week in order to schedule my first observation and interview. We can schedule both of them at that time, if you would like, or we can wait to schedule the second observation and interview after the first is complete.

Does anyone have any questions about what I am asking to do or what I am asking of you?

[Answer questions]

Please read and sign the consent form and return it to me. If you would like more time to read the consent form before deciding to sign it or not, I can leave it with you this morning and return this afternoon to collect it from you.

[Pass out forms]

Thank you very much for your time. To those of you who agree to participate in the study, I will email you in one week to schedule our observations. Goodbye.

APPENDIX H

Classroom Observation Protocol

Background information

Teacher name: _____ Date: _____

School: _____ Content Area(s): _____

Grade level: _____ Number of students: _____

Start time: _____ End Time: _____

Classroom description

In the space below, describe the classroom layout (e.g., seating arrangements, lab space vs. classroom space), students (e.g., number, gender, ethnicity) and teacher. Use drawings if necessary.

During observation

In this section you will describe what is occurring during each 5 minute segment of the observation. Use the codebook to fill in the codes for each category. Only indicate the activities that are occurring during the time. Because of the integrated and iterative nature of instruction, you may have more than one item in each category during each time period. In the notes indicate any interesting activity that may inform the overall description of the lesson below.

Time	CT Elements	Integration	Grouping	Notes
0-5	□Ab □G □Dc □Al □Db □CT	□Imp □Exp	□WG □SG □Ind	
5-10	□Ab □G □Dc □Al □Db □CT	□Imp □Exp	□WG □SG □Ind	
10-15	□Ab □G □Dc □Al □Db □CT	□Imp □Exp	□WG □SG □Ind	

15-20	□Ab □G □Dc □Al □Db □CT	□Imp □Exp	□WG □SG □Ind	
20-25	□Ab □G □Dc □Al □Db □CT	□Imp □Exp	□WG □SG □Ind	
25-30	□Ab □G □Dc □Al □Db □CT	□Imp □Exp	□WG □SG □Ind	
30-35	□Ab □G □Dc □Al □Db □CT	□Imp □Exp	□WG □SG □Ind	
35-40	□Ab □G □Dc □Al □Db □CT	□Imp □Exp	□WG □SG □Ind	
40-45	□Ab □G □Dc □Al □Db □CT	□Imp □Exp	□WG □SG □Ind	
45-50	□Ab □G □Dc □Al □Db □CT	□Imp □Exp	□WG □SG □Ind	
50-55	□Ab □G □Dc □Al □Db □CT	□Imp □Exp	□WG □SG □Ind	
55-60	□Ab □G □Dc □Al □Db □CT	□Imp □Exp	□WG □SG □Ind	
60-65	□Ab □G □Dc □Al □Db □CT	□Imp □Exp	□WG □SG □Ind	
65-70	□Ab □G □Dc □Al □Db □CT	□Imp □Exp	□WG □SG □Ind	
70-75	□Ab □G □Dc □Al □Db □CT	□Imp □Exp	□WG □SG □Ind	

75-80	□Ab □G □Dc □Al □Db □CT	□Imp □Exp	□WG □SG □Ind	
80-85	□Ab □G □Dc □Al □Db □CT	□Imp □Exp	□WG □SG □Ind	
85-90	□Ab □G □Dc □Al □Db □CT	□Imp □Exp	□WG □SG □Ind	

Lesson Description

In this section provide descriptive details about the lesson. Focus on descriptions of student-student and student-teacher interactions, specifics about integrated topics and instructional strategies and support mechanisms used by teacher.

Resources Review

In this section, review resources provided by the teacher that were used in planning or instruction of the lesson observed.

Name: _____ Type: _____

Source of Resource: □ Ind □ Dv Sp □ Sch Sp Medium: _____

CT Elements: □Ab □G □Dc □Al □Db □CT Integration: □Imp □Exp

Resource Description (including examples of CT and the elements of CT, strengths, weaknesses, etc.)

APPENDIX I
Informed Consent Agreement

Purpose of the Research Study: The purpose of this study is to understand the ways that teachers at Rusty Falls Elementary School teach computational thinking and the ways that they make use of resources to plan their instruction of computational thinking.

Procedures: This study will take place from October through December 2018. It will follow teachers in 3rd grade through two lessons integrating new curriculum. It will consist of observations of classroom instruction (approximately 60-90 minutes), collection of lesson-related documents, and post-observation interviews with teachers (approximately 60 minutes).

Risks: There are no anticipated risks in this study.

Benefits: There are no direct benefits to participating in this study. The recommendations of the researcher, however, intend to help administrators, teachers, and the district to become more knowledgeable about computational thinking and hopefully to improve its integration into instruction.

Cost of Participation: There is no cost to participate in this study.

Confidentiality: To ensure the confidentiality of the participants, all personally identifiable information will be removed, and pseudonyms will be used for persons and the research site. All data related to the study will be stored in password-protected files on a password-protected computer. At the conclusion of the study, the data will be stored on a password-protected file on an external hard drive that will be housed in a fire-safe box.

Voluntary Participation: Your participation in this study is voluntary and you are free to decide whether or not you would like to participate.

Right to Withdraw: You have the right to withdraw from the study at any time without penalty. If you choose to withdraw, all study materials related to you will be destroyed.

Compensation: You will not receive any compensation for participating in this study.

Questions: If you have questions, please contact:

Albert H. Jacoby, III	Jennifer Chiu, Ph.D.
University of Virginia	University of Virginia
Curry School of Education	Curry School of Education
Charlottesville, VA 22903	Charlottesville, VA 22903
434-242-2918	434-924-3915
ahj2yc@virginia.edu	jlchiu@virginia.edu

Go on to the next page

If you have questions about your rights in the study, contact:
Tonya R. Moon, Ph.D.
Chair, Institutional Review Board for the Social and Behavioral Sciences
One Morton Dr. Suite 500
University of Virginia, P.O. Box 800392
Charlottesville, VA 22908-0392
Telephone: (434) 924-5999
Email: irbsbshelp@virginia.edu
Website: www.virginia.edu/vpr/irb/sbs

Before you sign this form, please ask questions about any part of this study and/or consent that is not clear to you. Your signature below means that you have received this information and all of your questions have been answered.

You will receive a copy of this document after you have signed it.

Consent from Adult: I agree to participate in the research study described above

Participant (signature)	
Participant (print)	
Date	

APPENDIX J

Password Guidelines from Susie Jones's Second Observation

	Do	Do Not
1	Share password with parents	Share with others
2	Password has 8 characters	Use easy words about you that would be easy to guess
3	Use a combination of letters, numbers, symbols	Use private information in your password
4	Change your password every 6 months	

APPENDIX K

Smart Passwords?

Directions

Read the stories about Jessie and Krystal below and answer questions about their passwords.

Jesse lives in Lawrence, Kansas. He has a pet rat named "Phil" and is a big fan of the Kansas Jayhawks men's basketball team. Jesse chose "jayhawks" as his password.

Did he make a safe choice? Why or why not?

_____

_____

_____

Krystal lives in Miami, Florida. Her birthday is August 4, 1984, and she swims on a team. Her password is "krswim84."

How did Krystal choose her password? Was it a safe choice? Why or why not?

_____

_____

_____

You Try It!

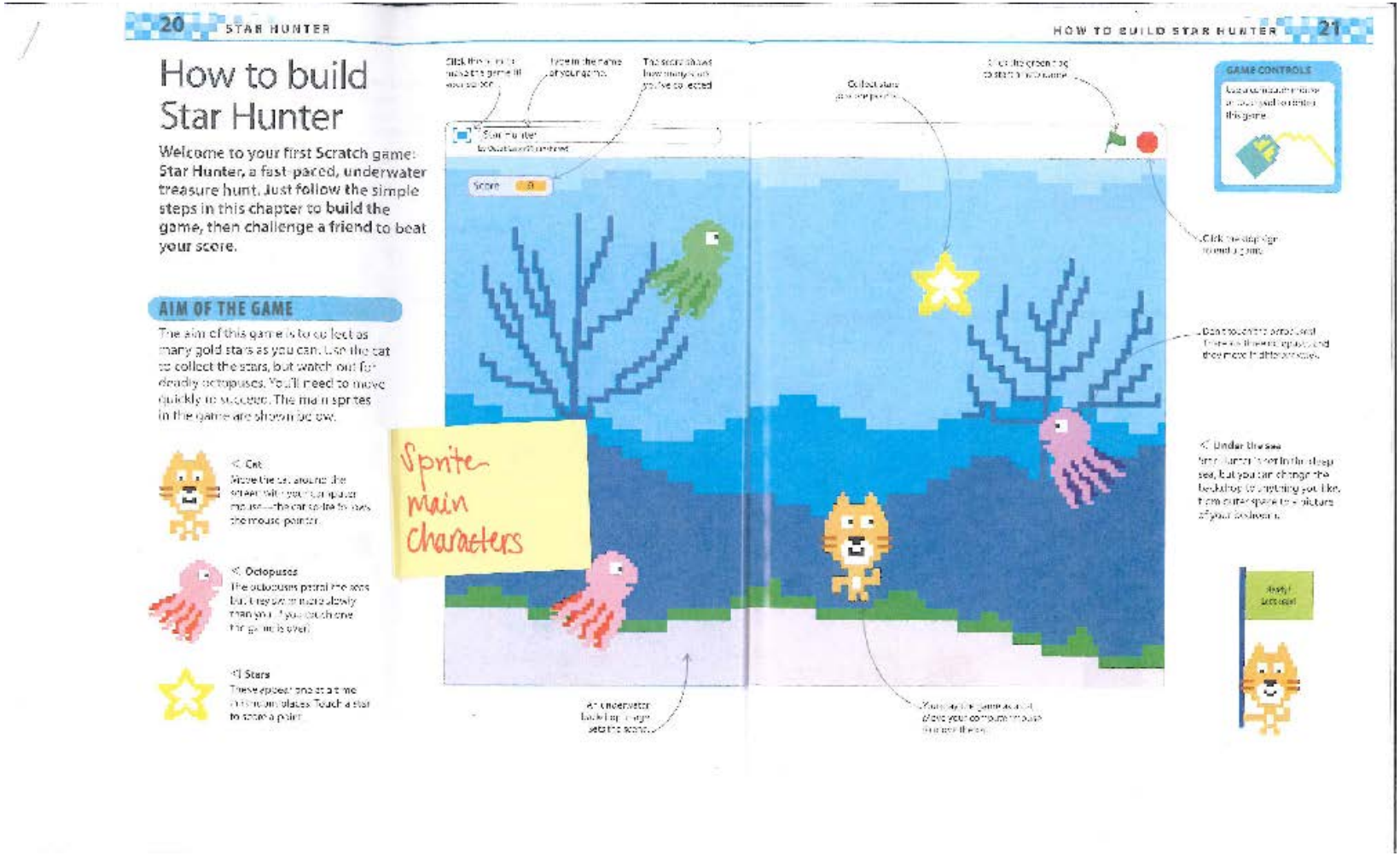
Use the Dos and DON'Ts tips to make new passwords for Jesse and Krystal.

Jesse _____

Krystal _____

APPENDIX L

Excerpt from *Coding Games in Scratch* (Woodcock, 2016)



20 STAR HUNTER

How to build Star Hunter

Welcome to your first Scratch game: Star Hunter, a fast-paced, underwater treasure hunt. Just follow the simple steps in this chapter to build the game, then challenge a friend to beat your score.

AIM OF THE GAME

The aim of this game is to collect as many gold stars as you can. Use the cat to collect the stars, but watch out for deadly octopuses. You'll need to move quickly to succeed. The main sprites in the game are shown below.

Cat
Move the cat around the screen with your computer mouse—the cat sprite follows the mouse pointer.

Octopuses
The octopuses patrol the seas but they can't move as fast as you. If you touch one the game is over.

Stars
These appear one at a time in random places. Touch a star to score a point.

Sprite main characters

HOW TO BUILD STAR HUNTER 21

GAME CONTROLS

Collect stars

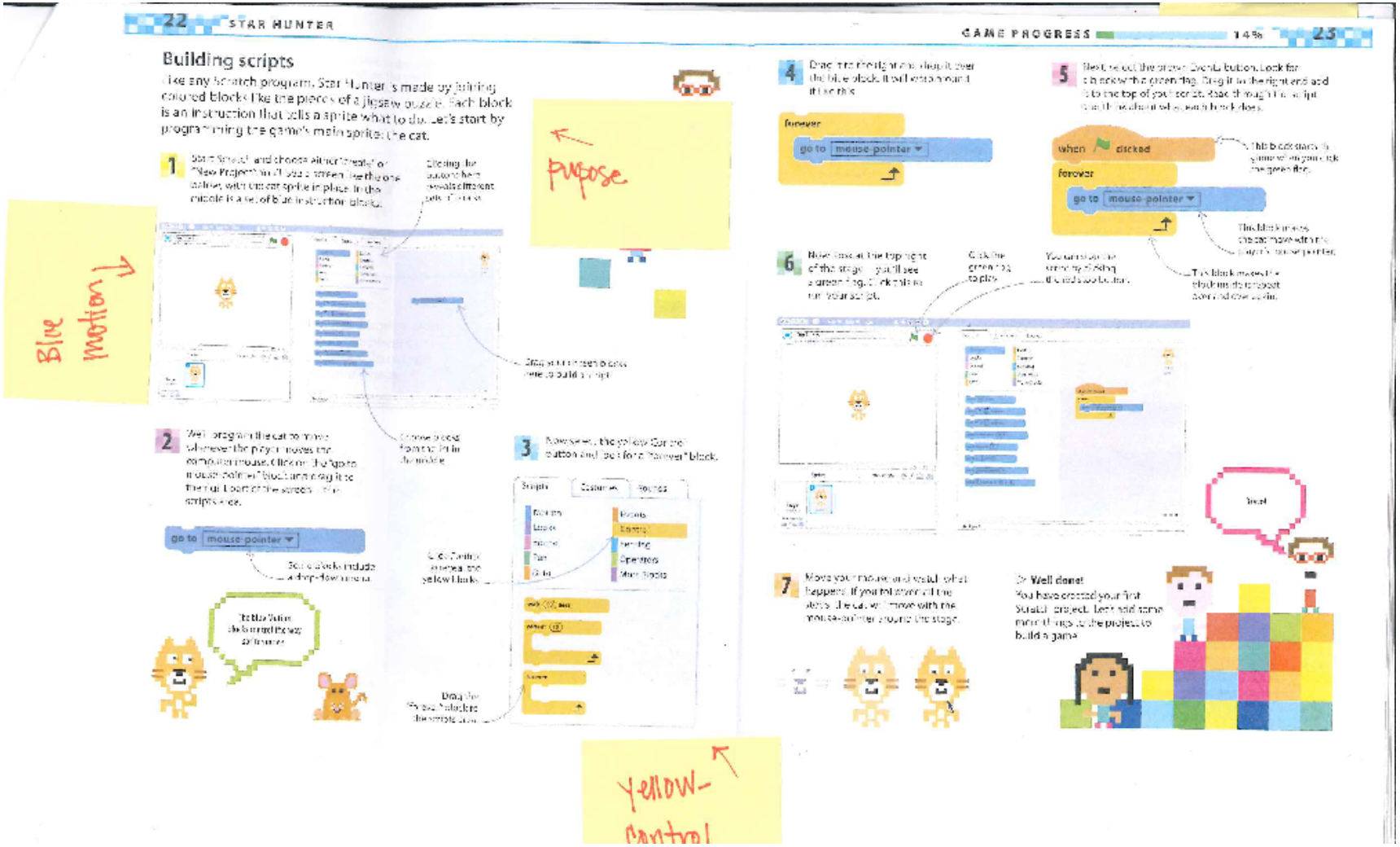
Click the green flag to start the game

Click the stop sign to end the game

Don't touch the octopuses!

Under the sea

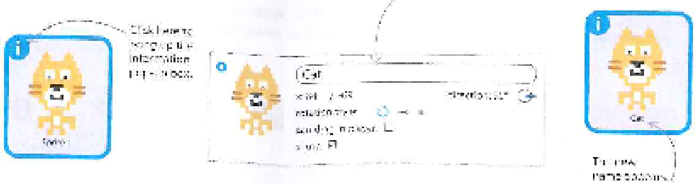
Ready! Set! Go!


22 STAR HUNTER
GAME PROGRESS 14%
23
Building scripts
Like any Scratch program, Star Hunter is made up joining coloured blocks like the pieces of a jigsaw puzzle. Each block is an instruction that tells a sprite what to do. Let's start by programming the game's main sprite, the cat.
1
2
3
4
5
6
7
go to mouse-pointer
forever
when clicked
Well done!
purpose
Blue motion
yellow- control

adding details to sprite

back-ground library

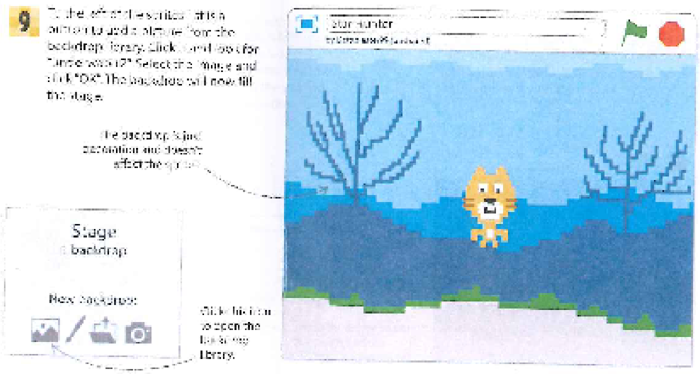
8 The cat is called "Sprite1." Let's fix that. In the sprites list, select Sprite1 (the cat) and click on the blue "i" in the corner to get more information about the sprite. Change the name to "Cat."



Setting the scene

At the moment, the stage is just a boring white rectangle. Let's create some atmosphere by adding scenery and sound effects. To change the scenery, we add a "backdrop" image.

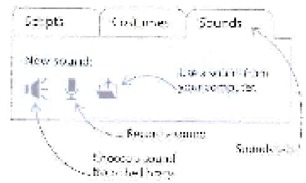
9 To the left of the sprites list is a button to add a picture from the backdrop library. Click it and look for "underwater2." Select the image and click "OK." The backdrop will now fill the stage.



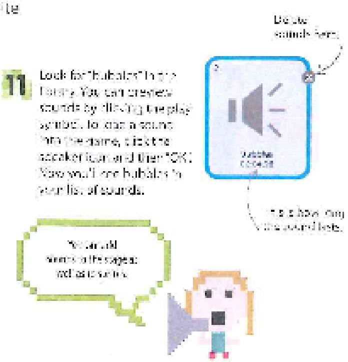
Sound effects

Now we'll add a bubbling sound to the cat sprite to make it sound like we're underwater.

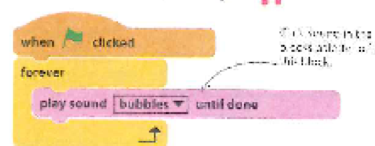
10 Highlight the cat in the sprites list and then click the Sounds tab above the blocks palette. Click the speaker icon to choose a sound from the library.



11 Look for "bubbles" in the library. You can preview sounds by clicking the play symbol. To load a sound into the game, click the speaker icon and then "OK." Now you'll see bubbles in your list of sounds.



12 Click the Scripts tab and add the following script to the cat sprite, but leave the old script in place because you need both. The new script repeats the bubbles sound. The "play sound ... until done" block waits for the sound to finish before letting it start again. Run the game to hear the sound effect.



EXPERT TIPS

Loops

A loop is a section of code that repeats over and over again. The "forever" block creates a loop that carries on forever, but other types of loop can repeat an action a fixed number of times. Loops are very common in almost all computer programming languages.

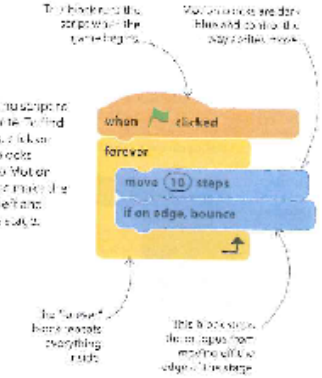
Pink ↑ Sound

Add an enemy

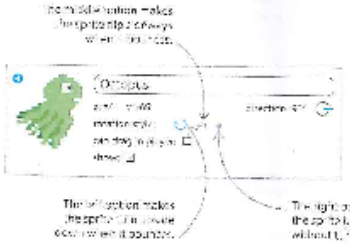
The game needs an enemy to make things more interesting. Let's add an octopus with a deadly sting. The octopus will patrol the stage, moving left and right, and the player will have to keep out of its way or the game's over.

13 To add a second sprite to the stage, click the icon shown below to open up the sprite library. Choose the octopus and click OK.

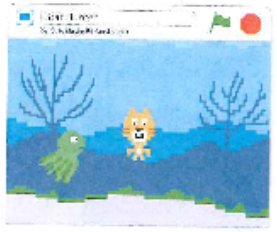
14 Add the following octopus code. The octopus sprite is found in the blue blocks (click Motion in the blocks palette). The two Motion blocks used here make the octopus move left and right across the stage.



15 Now run the script. The octopus will patrol left to right, but you'll notice it sometimes does it the wrong way around. We can fix this by changing the way the sprite turns around when it reaches the edge. Right-click the octopus sprite in the Sprite pane. In the pop-up box, there are three options after "rotation style".



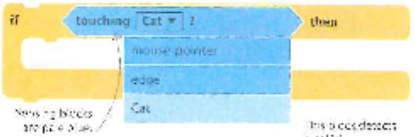
16 Choose the middle option and run the program. The octopus should now turn right, slide up and start facing forward all the time. You can adjust the starting position of the sprite by dragging it with the mouse.



Collisions

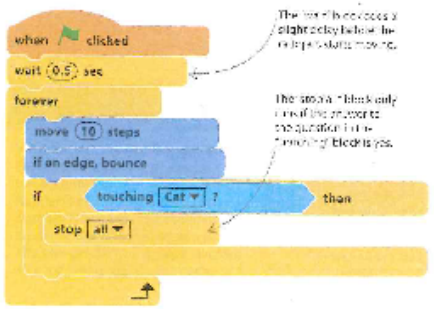
So far the octopus and cat move through each other without anything happening. We need to add a script to make them stop moving when they collide. Collision detection is very important in computer games.

17 Highlight the octopus and drag a yellow "if then" block to an empty part of the scripts area. Now add a sprite value from the top of the "if then" block. Click the drop-down menu and choose Cat. This script will help the octopus detect the cat.



18 Choose Control in the blocks palette again and add a stop all block to the middle of the "if then" block. This action will stop the octopus touching the cat ending the game.

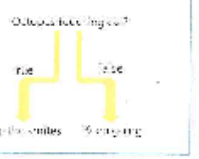
19 Now add the "if then" blocks you've built to the octopus's main script, placing it carefully after the Motion blocks. Also, add a "wait 0.5 sec" before the loop. Run the project and see what happens.



EXPERT TIPS

"if then"

You make decisions every day. If it's raining, you might use an umbrella. If it isn't, you don't. Computer programs do the same thing by using what programmers call conditional statements. Such as "if then". When Scratch reaches an "if then" block, it runs the blocks inside only if the statement is true.



Sprite library
Myw Code break

Other examples of collision in game

APPENDIX M

Code.org Unplugged Lesson on Computational Thinking

U

C O
D E

UNPLUGGED

Computational Thinking

Lesson time: 25 Minutes Basic lesson time includes activity only. Introductory and Wrap-Up suggestions can be used to delve deeper when time allows.

LESSON OVERVIEW

For this activity, no instructions are provided. Instead, students will use examples of what imaginary players have done to figure out how to play the game. This lesson gives students the opportunity to practice the four arts of computational thinking (decomposition, pattern matching, abstraction, and algorithms) in one cohesive activity.

TEACHING SUMMARY

Getting Started - 15 minutes

1) Vocabulary
2) Figuring it Out

Activity: Computational Thinking - 25 minutes

3) Computational Thinking

Wrap-up - 10 minutes

4) Flash Chat - What did we learn? 5) Vocab Shmocab

Assessment - 5 minutes

6) Computational Thinking Assessment

LESSON OBJECTIVES

Students will:

- Analyze information to draw conclusions
- Match identical portions of similar phrases to match patterns
- Identify differences in similar phrases and abstract them out

TEACHING GUIDE

MATERIALS, RESOURCES AND PREP

For the Student

- One die per group
- One Computational Thinking Kit per group
- Pens, Pencils, & Scissors
- Computational Thinking Assessment for each student

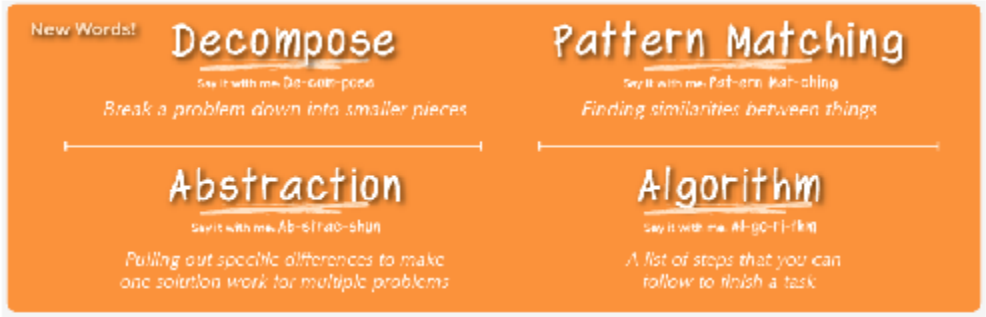
For the Teacher

- Lesson Video
- This Teacher Lesson Guide
- Print one Computational Thinking Kit per group
- Print one Computational Thinking Assessment for each student

GETTING STARTED (15 MIN)

1) Vocabulary

This lesson has four new and important words:



Algorithm - Say it with me: Al-go-ri-thm
A list of steps that you can follow to finish a task

Decompose - Say it with me: De-com-pose
Break a problem down into smaller pieces

Abstraction - Say it with me: Ab-strac-shun
Pulling out specific differences to make one solution work for multiple problems

Pattern Matching - Say it with me: Pat-ern Matching
Finding similarities between things

2) Figuring it Out

- Tell your students that you want them to sum up all of the numbers between 1 & 200.
  - Use your body language to indicate that this is not a "serious" or graded exercise.
    - Now, let them know that they must do it all in their heads.
    - Add the time constraint of thirty seconds.
    - They may feel overwhelmed. This is intentional. You can indicate with your tone and demeanor that you might be crazy asking this of them, but begin timing with a resounding: "Starting NOW".
- Watch the class as you keep time. How many are lost in thought?
- When time is up, ask if anyone was able to get the total.
- Ask if there is anyone who thought the problem was so hard that they didn't even attempt it.
- Did anyone attempt it and just not finish?
  - What did they try?
- Guide students toward thinking a little smaller.
  - If we break the problem up into smaller pieces, it becomes easier to manage.
  - Let's start at the two ends. What is 200 + 1?
  - What is 199 + 2?
  - What is 198 + 3?
  - See a pattern?
  - How many of these pairs will we have?
    - What is the last pair we will find? 100 + 101
    - That means that we have 100 total pairs.

        - If we have 100 total pairs of sums of 201, how do we find the final total?
        - What is 100 * 201?
            - Now, what if we wanted to find the trick to do this with other numbers?
    - Can we do it easily with 2,000?
    - How about 20,000?
    - What stays the same? What is different?
    - If we use abstractions to make our end goal something that can change (say we name it "blank") then we can make an algorithm that will work for any number
- Work through the problem until you ultimately get ? = ("blank"/2) * ("blank"+1)
- Do a few simple examples to show that the algorithm is correct for blanks= 2, 3, 4, & 5.

"This is all to show that if you use the tools of Computational Thinking (decomposition, pattern matching, abstraction, and algorithms), then you can figure out how to solve problems that no one has already taught you how to solve...just like we did here! This will be an extremely powerful skill for the rest of your life!"

ACTIVITIES: (25 MIN)

3) Computational Thinking

This lesson is all about a "Game with No Instructions." Students will be charged with figuring out how to play the game as a small group. The small details of their final algorithm are unimportant. What *is* important is that they were able to take a huge task like "figuring out how to play a game on their own" and take small steps toward achieving the goal.

Students will be guided toward discovering the rules using the steps of computational thinking. Resist the temptation to point the students toward "doing it right" and allow them just to do it on their own. If they feel stumped or confused, encourage the students to look at the information that has been given to them, or if they must, ask a classmate.

Directions:

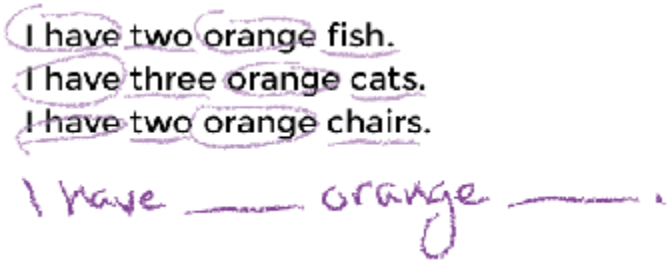
1) Divide students into groups of 2-4.

2) Have the groups read over user experiences to get an idea of how other students have played the "Game with No Instructions."

3) Encourage them to pattern match between each experience by circling the sections of words that are identical from player to player.

4) Next, have them abstract away differences from each experience by underlining words that change from player to player.

5) Using pattern matching and abstraction, have them make a script template for game play by writing up the circled parts of the other students' experiences, and leaving the underlined sections as blanks.



For example:

6) Give students a blank sheet of paper to write a list of instructions for how they think this game should be played based on the user experiences that they just read. This will be their algorithm.

7) Have students play the game using the algorithm that they just made. Each player should get at least two turns.

WRAP-UP (5 MIN)

4) Flash Chat: What did we learn?

- What should you try to do when you're asked to do something and you don't know how?
- If a problem is too hard, what should you try to do?
- If you find similarities in lots of solutions to different problems, what does that probably tell you?
- If you have a problem that is just a little different from a problem that you have a solution for, what would you do?

LESSON TIP

Flash Chat questions are intended to spark big-picture thinking about how the lesson relates to the greater world and the students' greater future. Use your knowledge of your classroom to decide if you want to discuss these as a class, in groups, or with an elbow partner.

5) Vocab Shmocab

- Which one of these definitions did we learn a word for today?

"Bringing two pieces together"
"Breaking a problem down into smaller pieces"
"An educated guess"

...and what is the word that we learned?

ASSESSMENT (5 MIN)

6) Computational Thinking Assessment

- Hand out the assessment worksheet and allow students to complete the activity independently after the instructions have been well explained.
- This should feel familiar, thanks to the previous activities.

U

Unplugged

Name: _____ Date: _____

Computational Thinking

User Experience Scripts

C O D E

Figure out how to play this game by looking at the players' phrases below. Circle the matching parts and underline words that are different from player to player. The first matching section has been circled for you.

Player 1:

"I chose a lion, and rolled a six, then a four, then a two. That means I need to draw a black cupcake on my lion's tail."

Player 2:

"I chose a donkey, and rolled a three, then a two, then a one. That means I need to draw a yellow pineapple on my donkey's head."

Player 3:

"I chose a puppy, and rolled a five, then a three, then a five. That means I need to draw a pink salmon on my puppy's nose."

Using pattern matching and abstraction, make yourself a template for game play by writing up the circled parts of the other students' experiences, and leaving the underlined sections as blanks.

"I chose a _____

_____

_____

_____

_____

Computational Thinking

Roll Catalog

Color:

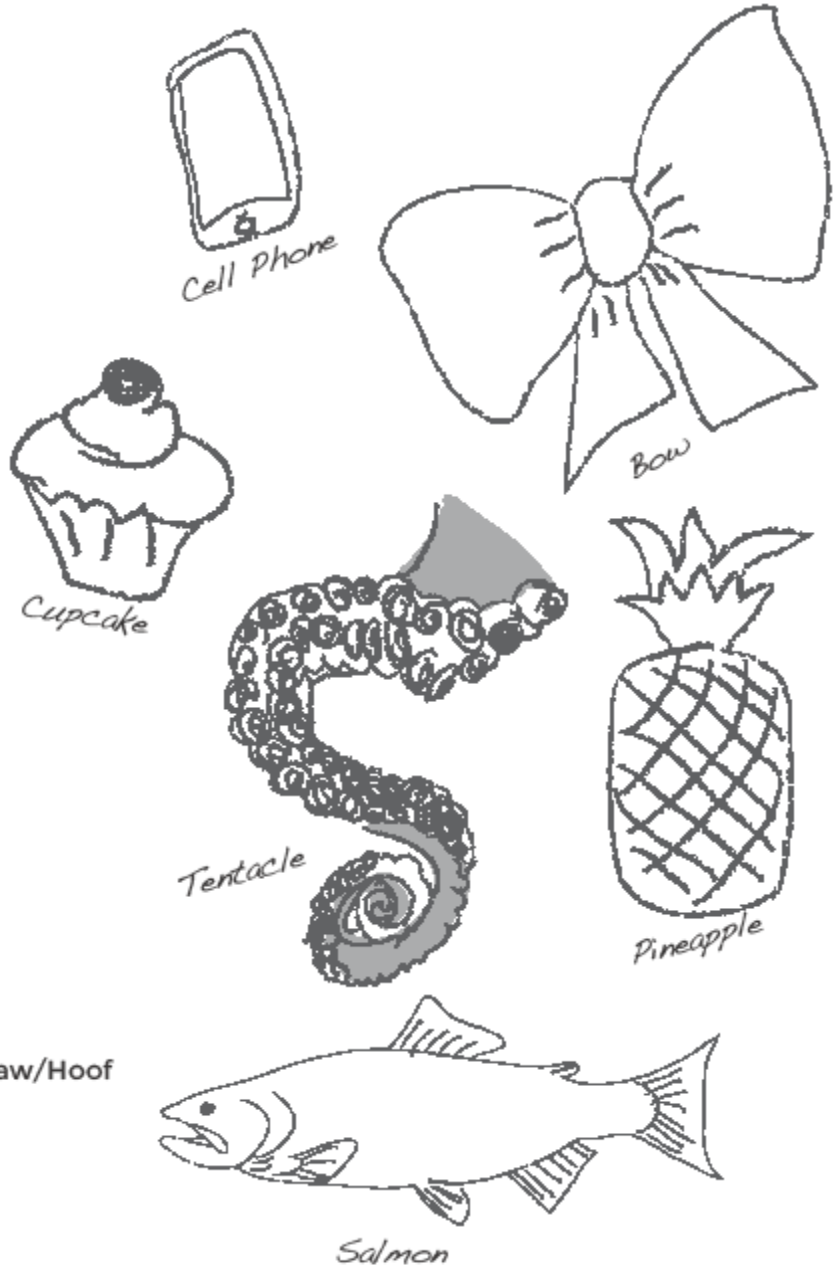
1) Red
2) Blue
3) Yellow
4) Green
5) Pink
6) Black

Items:

1) Cell Phone
2) Pineapple
3) Salmon
4) Cupcake
5) Tentacle
6) Bow

Part:

1) Head
2) Tail
3) Hand/Foot/Paw/Hoof
4) Belly
5) Nose
6) Back

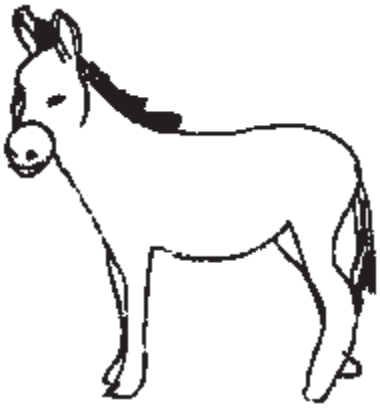


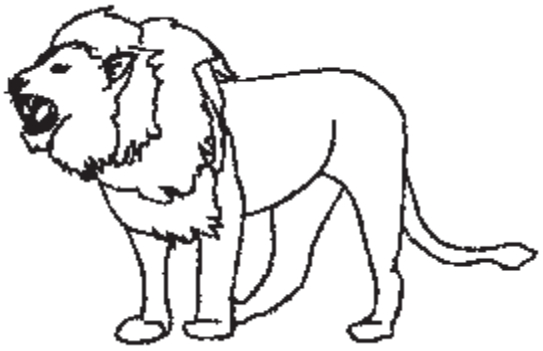
U

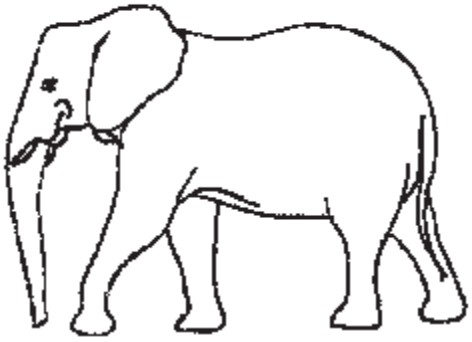
Computational Thinking

Decomposition, Patterns, Abstraction, Algorithms

C O D E







APPENDIX N

Computer Technology Standards of Learning – Grades 3-5

Introduction

> *As the new century has unfolded, various studies have postulated about the likely competencies that will be needed in the workplace of tomorrow; one consistent conclusion is that technology will be integrated into every facet of business and life.*
>
> *The* Educational Technology Plan for Virginia: 2010-15 *focuses primarily on one specific component of 21st century skills—information and communications technology (ICT) literacy. The most recognized definition for this topic was formulated in 2002 by the International ICT Literacy Panel: "ICT literacy is using digital technology, communications tools, and/or networks to access, manage, integrate, evaluate, and create information in order to function in a knowledge society."*
>
> *Educational Technology Plan for Virginia: 2010-15*

The Computer Technology Standards of Learning define the essential knowledge and skills necessary for students to access, manage, evaluate, use, and create information responsibly using technology and digital resources. They provide a framework for digital literacy and include the progressive development of technical knowledge and skills, intellectual skills for thinking about and using information, and skills needed for working responsibly and productively both individually and within groups. Digital literacy is not an end in itself but lays the foundation for deep and continuous learning. It focuses on using technology to learn rather than learning about technology.

To become technologically proficient, students must develop these skills through integrated activities across all K-12 content areas. These skills should be introduced and refined collaboratively by all K-12 teachers as an integral part of the learning process. Teachers can use these standards as guidelines for planning technology-based activities in which students achieve success in learning and communication—preparing them to meet the challenges of today's knowledge-based society.

Grades 3-5

Basic Operations and Concepts

C/T 3-5.1 Demonstrate an operational knowledge of various technologies.

A. Use various types of technology devices to perform learning tasks.
- Use a keyboard, mouse, touchscreen, touchpad, and other input devices to interact with a computer.
- Demonstrate the ability to perform a wide variety of basic tasks using technology, including saving, editing, printing, viewing, and graphing.

B. Communicate about technology with appropriate terminology.
- Use basic technology vocabulary in daily practice.

C/T 3-5.2 Identify and use available technologies to complete specific tasks.

A. Identify the specific uses for various types of technology and digital resources.
- Identify the differences among local, network, and Internet resources and tools.
- Create, edit, and format a document with text and graphics.
- Create and present a multimedia presentation.
- Create and populate a spreadsheet with data.
- Capture and edit a digital image.
- Demonstrate the ability to choose appropriate resources when completing assignments in various content areas.

B. Use content-specific tools, software, and simulations to complete projects.
- Use tools in various content areas as directed by the teacher.

Social and Ethical Issues

C/T 3-5.3 Make responsible decisions—grounded in knowledge of digital safety and security best practices—that pertain to various digital communication tools and methods.

A. Demonstrate knowledge of basic practices related to online safety.
- Use best practices for online safety as defined by the division's online safety program.
- Demonstrate an understanding of the division's acceptable use policy and consequences for inappropriate use.

B. Discuss and model responsible behaviors when using information and technology.
- Identify reasons for taking security precautions when using any technology, especially those related to the Internet.
- Demonstrate responsible behavior, such as using strong passwords and
avoiding high-risk activities.

- Identify inappropriate or threatening interpersonal situations involving electronic devices and develop strategies to react to them safely.

- Behave appropriately in virtual groups and be proactive in preventing bullying behavior in an environment that provides anonymity to bullies.

C/T 3-5.4 Exhibit personal responsibility for appropriate, legal, and ethical conduct.

A. Understand the need for laws and regulations regarding technology use.
   - Model appropriate, legal, and ethical behavior in all technology use and technology-supported environments.

B. Understand the basic principles of the ownership of ideas.
   - Demonstrate a basic understanding of "fair use."

C/T 3-5.5 Demonstrate digital citizenship by actively participating in positive activities for personal and community well-being.

A. Communicate respect for people when participating in group online learning activities.
   - Identify ways in which online communications are different from face-to-face communications.
   - Demonstrate online etiquette when communicating with others.

B. Explore the potential of the Internet as a means of personal learning and the respectful exchange of ideas and products.
   - Participate in the creation of digital projects that involve communicating with others.

Technology Research Tools

C/T 3-5.6 Plan and apply strategies for gathering information, using a variety of tools and sources, and reflect on alternate strategies that might lead to greater successes in future projects.

A. Collect information from a variety of sources.
   - Conduct research using various types of text- and media-based information.

B. Apply best practices for searching digital resources.
   - Apply effective search strategies that will yield targeted information.
   - Identify basic indicators that a digital source is likely to be reliable.

C/T 3-5.7 Draw conclusions from research and relate these findings to real-world situations.

A. Use research to support written and oral presentations.
   - Apply research derived from digital resources to original work.
   - Demonstrate how to cite digital resources when developing nonfiction reports and presentations.

B. Apply knowledge when conducting research to develop accurate and balanced reports.
- Use best practice guidelines for evaluating research results.

Thinking Skills, Problem Solving, and Decision Making

C/T 3-5.8 Practice reasoning skills when gathering and evaluating data.

A. Determine when technology tools are appropriate to solve a problem and make a decision.
- Identify technology resources and tools that can help with decision making.

B. Demonstrate organization and persistence when completing personal and group assignments, activities, and projects.
- Use various productivity tools that help with planning, time management, project goal setting, etc.

C/T 3-5.9 Use models and simulations to understand complex systems and processes.

A. Understand the use of simulations in learning.
- Enhance understanding of concepts and skills by explaining how a simulation differs from and is similar to real life.

B. Use simulations to understand complex concepts.
- Enhance understanding of concepts and skills by using simulations.

Technology Communication Tools

C/T 3-5.10 Communicate effectively with others (e.g., peers, teachers, experts) in collaborative learning situations.

A. Use technology tools for individual and collaborative writing, communication, and publishing activities.
- Produce documents and presentations that demonstrate the ability to edit, reformat, and integrate various tools and media.

B. Participate in communications among different cultures.
- Understand the need to place communication in the context of culture.

C. Assume different roles (e.g., leader/follower, orator/listener) on teams in various situations.
- Recognize that different people on a team bring different technical skills, and understand how that can influence team responsibilities.
- Demonstrate the ability to share technology tools as needed.

C/T 3-5.11 Apply knowledge and skills to generate innovative ideas, products, processes, and solutions.

A. Organize and display knowledge and understanding in ways that others can view, use, and assess.
- Understand the various ways in which digital products can be shared.

B. Use technology tools to share original work.
   - Use presentation tools to organize and present stories, poems, songs, and other original work.

APPENDIX O

Create-Your-Own-Stories ROUGH DRAFT

Students will figure out how to play your game by looking at the players' phrases below. They will circle the matching parts and underline words that are different from player to player. The first matching section will be circled for them.

Player 1

I chose _____ . I rolled a _____, _____, and

(pick an item from the last page) (number on dice), (number on dice),

_____ so I _____ with _____ and

(number on dice) (dice roll 1 category item) (dice roll 2 category item)

_____.

(dice roll 3 category item)

Player 2

I chose _____ . I rolled a _____, _____, and

(pick an item from the last page) (number on dice), (number on dice),

_____ so I _____ with _____ and

(number on dice) (dice roll 1 category item) (dice roll 2 category item)

_____.

(dice roll 3 category item)

Player 3

I chose _____ . I rolled a _____, _____, and

(pick an item from the last page) (number on dice), (number on dice),

_____ so I _____ with _____ and

(number on dice) (dice roll 1 category item) (dice roll 2 category item)

_____.

(dice roll 3 category item)

Students will then use pattern matching and abstraction, make themselves a template for game play by writing up the circled parts of the other students' experiences, and leaving the underlined sections as blanks. What are you going to be looking to see that the students can do?

_____

_____

_____

_____

Draw some images here

Characters:

1)

2)

3)

4)

5)

6)

Setting:

1)

2)

3)

4)

5)

6)

Problem:

1)

2)

3)

4)

5)

6)

Solution:

1)

2)

3)

4)

5)

6)

Extra (Optional):
Here is where you can add magical characteristics or unusual creatures or whatever!

1)

2)

3)

4)

5)

Draw 6 objects that your friends can choose from with no instruction included.